# STPC® Elite Programming Manual

Issue 1.1

February 5, 2002

**STMicroelectronics**

# 1 Table of Contents

## Table of Contents

## 2 List of Tables

## List of Tables

# 3 List of Figures

## List of Figures

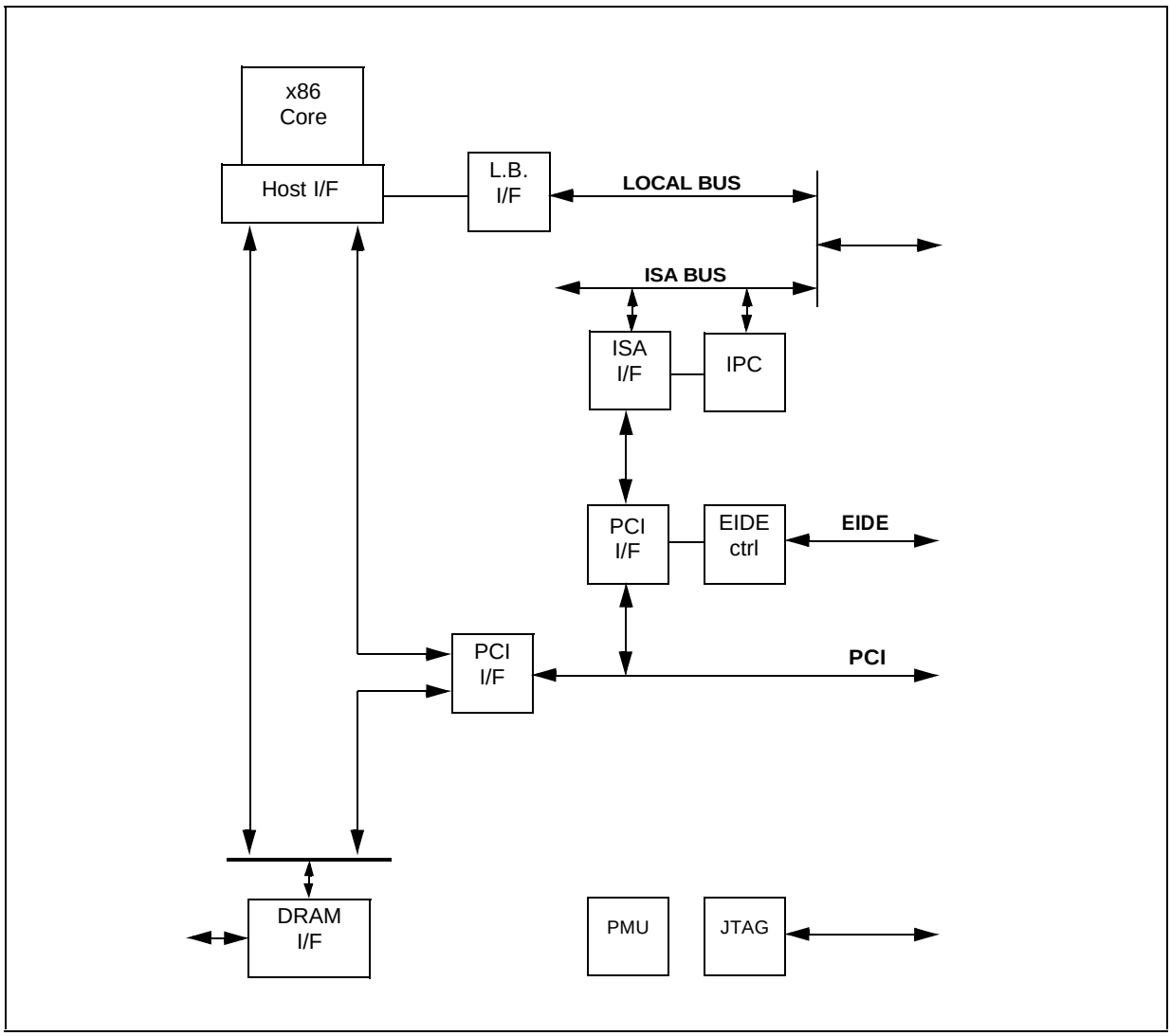
# 4. INTRODUCTION

This programming manual details the register sets for the STPC Elite Device. The manual is split into chapters each dedicated to a function.

This document contains all the information required to program and configure the STPC Elite.

In order to use this manual to the full, you may want to make reference to the STPC and X86 core Datasheets.

**Figure 4-1. Functionnal description**



# 5. HOW TO USE THIS MANUAL

## 5.1. INTRODUCTION

This manual provides full technical documentation for the STPC device. It is recommended that the reader is familiar with the x86 series processors and PC compatible architectures before reading this document. Many terms are related directly to the PC architecture.

The manual itself is split into chapters. These chapters hold the information for a particular functional block of the device. For example, the chapter titled “Memory Access” gives the memory map of the STPC device, the memory architecture and interface to the external DRAM modules.

## 5.2. SPECIFIC NOTES

### *5.2.1. RESERVED BITS*

Write mode 1 is a subset of Write Mode 0. No CPU-supplied write data is used. The read data latched from a previous read operation is written. The bit mask is disabled. The map-masks are implemented as they are for Write Mode 0.

Many bits in the register descriptions are noted as reserved. These bits are not internally connected, physically not present or are used for testing purposes. In all cases these bits should be set to a ‘0’ when writing to a register with reserved bits. When reading from a register with reserved bits, these specific bits should be masked from the data value before action is taken on the data.

Any functionality found by setting the reserved bits to levels other than ‘0’ cannot and will not be guaranteed on future revisions of the circuit design. Thus it is not recommended to use the bits marked as reserved in any way different from noted above.

### *5.2.2. SIGNAL ACTIVE STATE*

The hash symbol (#) following a signal name indicates that when the signal is in its active (asserted) state, the signal is at a logic low level. When the “#” is not present at the end of a signal name, the logic high level represents the active state.

### *5.2.3. HEXADECIMAL NOTATION*

In this manual Hexadecimal (Hex) numbers (numbers to the base 16: [0-9,A-F]) are denoted by the postfix ‘h’.

For example a memory address 783A hexadecimal will be written 783Ah.

### *5.2.4. ENDIAN*

In common with the x86 architecture, values in memory are little-endian, that is the lower part of the memory contains the least significant Byte.

For an 8-bit value

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| N | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

For a 16-bit (word) value

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| N | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| N+1 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |

For a 24-bit value

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| N | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| N+1 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| N+2 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |

For a 32-bit (long word) value

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| N | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| N+1 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| N+2 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| N+3 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |

For a 64-bit (QUAD word) value

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| N | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| N+1 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| N+2 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| N+3 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| N+4 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| N+5 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 |
| N+6 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 |
| N+7 | 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 |

### 5.3. ISSUING NOTES

There are three levels identified; Advanced data, Preliminary data and Full production release.

Each level is identified in a specific way as follows.

| Document Identification | Status | Definition | Release Identification |
|---|---|---|---|
| ADVANCED DATA | In design | This document based on the product specification. The information may be updated without notice. Large changes may still occur. | Release A, Release B... |
| PRELIMINARY DATA | Pre-production Data | This document contains preliminary data and may be updated without notice in order to improve the product features. | Issue 0.X. |
| FULL PRODUCTION DATA | Production Data | This is the finalised document and all test plans are completed. The information may be updated without notice in order to improve the product features. | Issue 1.X. |

# 6 LIST OF REGISTERS

This chapter lists all the registers accessable by external software.

| Section | Register Name | Mnemonic | Purpose | Address | Access type |
|---|---|---|---|---|---|
| 3. | **Power on strap Registers** | | | 022h/023h | |
| 3.1.1. | Strap register 0 | Strap0 | Configuration | | Index 04Ah |
| 3.1.2. | Strap Register 1 | Strap1 | Configuration | | Index 04Bh |
| 3.1.3. | Strap Register 2 | Strap2 | Configuration | | Index 04Ch |
| 3.1.4. | HCLK Strap register | HCLK_Strap | Configuration | | Index 05Fh |
| | | | | | |
| 7.5. | **Cache related registers** | | | 022h/023h | |
| 7.5.1. | Cache Architecture Register 0 | Cash_Arc0 | Configuration | | Index 020h |
| 7.5.2. | Cache Architecture Register 1 | Cash_Arc1 | Configuration | | Index 021h |
| 7.5.3. | Cache Architecture Register 2 | Cash_Arc2 | Configuration | | Index 022h |
| | | | | | |
| 7.6. | **Address decode related registers** | | | 022h/023h | |
| 7.6.1. | Memory Hole Control Register | MEM_HOLE | Configuration | | Index 024h |
| 7.6.2. | Shadow Control Register 0 | SHADOW_0 | Configuration | | Index 025h |
| 7.6.3. | Shadow Control Register 1 | SHADOW_1 | Configuration | | Index 026h |
| 7.6.4. | Shadow Control Register 2 | SHADOW_2 | Configuration | | Index 027h |
| 7.6.5. | Shadow Control Register 3 | SHADOW_3 | Configuration | | Index 028h |
| | | | | | |
| 7.7. | **Host SDRAM controller registers** | | | 022h/023h | |
| 7.7.1. | SDRAM Bank 0 Register | SDRAM_bank 0 | Configuration | | Index 030h |
| 7.7.2. | SDRAM Bank 1 Register | SDRAM_bank 1 | Configuration | | Index 031h |
| 7.7.3. | SDRAM Bank 2 Register | SDRAM_bank 2 | Configuration | | Index 032h |
| 7.7.4. | SDRAM Bank 3 Register | SDRAM_bank 3 | Configuration | | Index 033h |
| 7.7.5. | SDRAM Refresh Register | SDRAM_Ref | Configuration | | Index 039h |
| | | | | | |
| 8.3. | **SDRAM register access** | | | 84C6000h | |
| 8.3.1. | Register 0 | MEM_REG0 | Configuration | | 000h |
| 8.3.2. | Register 1 | MEM_REG1 | Configuration | | 004h |
| 8.3.3. | Register 2 | MEM_REG2 | Configuration | | 008h |
| 8.3.4. | DDC Control Register | DDCR | Configuration | 022h/023h | 95h |
| | | | | | |
| 8.4. | **MCLK Control Registers** | | | 022h/023h | |

## LIST OF REGISTERS

| Section | Register Name | Mnemonic | Purpose | Address | Access type |
|---|---|---|---|---|---|
| 8.4.1. | MCLK Control Register 0 | MCLK00 | | | Index 0x40h |
| 8.4.2. | MCLK Control Register 1 | MCLK01 | | | Index 0x41h |
| | | | | | |
| 9.5. | **North Bridge Configuration Registers** | | | | |
| 9.3. | Configuration Address Register | Config_ address | IO | 0xCF8h | |
| 9.4. | Configuration Data Register | Config_data | IO | 0xCFC - CFFh | |
| 9.5. | North Bridge Vendor Identification Register | NB_V_ID | PCI Config | | Index 0x0h |
| 9.5. | North Bridge Device Identification Register | NB_D_ID | PCI Config | | Index 0x2h |
| 9.5.1. | North Bridge PCI Command Register | NB_Com | PCI Config | | Index 0x4h |
| 9.5.2. | North Bridge PCI Status Register | NB_Stat | PCI Config | | Index 0x6h |
| 9.5.3. | North Bridge PCI Revision Id Register | NB_R_ID | PCI Config | | Index 0x8h |
| 9.5.4. | North Bridge Device Class Code Register | NB_C_ Code | PCI Config | | Index 0x9h |
| 9.5.5. | North Bridge Header Type Register | NB_Control | PCI Config | | Index 0x0Eh |
| 9.5.6. | North Bridge Control Register | NB_Cont | PCI Config | | Index 0x50h |
| 9.5.7. | North Bridge PCI Error Status Register | NB_E_ Stat | PCI Config | | Index 0x54h |
| | | | | | |
| 9.7. | **South Bridge PCI Function 0 Configuration Registers** | | | 0xCF8h | |
| 9.7. | South BridgeVendor Identification Register | SB_V_ID0 | PCI config F#0 | 0xCFCh | Index 0x0h |
| 9.7. | South Bridge Device Identification Register | SB_D_ ID0 | PCI Config F#0 | | Index 0x2h |
| 9.7.1. | South Bridge PCI Command Register | SB_ Com_0 | PCI Config F#0 | | Index 0x4h |
| 9.7.2. | South Bridge PCI Status Register | SB_Stat0 | PCI Config F#0 | | Index 0x6h |
| 9.7.3. | South Bridge PCI Revision Id Register | SB_R_ID0 | PCI Config F#0 | | Index 0x8h |
| 9.7.4. | South Bridge Device Class Code Register | SB_C_ Code0 | PCI Config F#0 | | Index 0x9h |
| 9.7.5. | South Bridge Header Type Register | SB_Head0 | PCI Config F#0 | | Index 0xEh |
| 9.7.6. | South Bridge Miscellaneous Register | SB_Misc0 | | | Index 040h |
| | | | | | |
| 9.8. | **South Bridge PCI Function 1 Configuraiton Registers** | | | 0xCF8h | |

| Section | Register Name | Mnemonic | Purpose | Address | Access type |
|---|---|---|---|---|---|
| 9.8.1. | South Bridge Vendor Identification Register | SB_V_ID1 | PCI config F#1 | 0xCFCh | Index 0x0h |
| 9.8.2. | South Bridge Device Identification Register | SB_D_ ID1 | PCI Config F#1 | | Index 0x2h |
| 9.8.3. | South Bridge PCI Command Register | SB_ Com1 | PCI Config F#1 | | Index 0x4h |
| 9.8.4. | South Bridge PCI Status Register | SB_Stat1 | PCI Config F#1 | | Index 0x6h |
| 9.8.5. | South Bridge Revision ID Register | SB_R_ID1 | PCI Config F#1 | | Index 0x8h |
| 9.8.6. | South Bridge Programming Interface Register | Prog_Int | PCI Config F#1 | | Index 0x9h |
| 9.8.7. | South Bridge Sub-Class Code Register | Sub_Class | PCI Config F#1 | | Index 0xAh |
| 9.8.8. | South Bridge Base-Class code Register | Base_Class | PCI Config F#1 | | Index 0xBh |
| 9.8.9. | South Bridge Latency Timer control Register | Lat_Timer | PCI Config F#1 | | Index 0xDh |
| 9.8.10. | South Bridge Header Type Register | Head_Time | PCI Config F#1 | | Index 0xEh |
| 9.8.11. | South Bridge Base Address 0 Register | Base0 | PCI Config F#1 | | Index 0x10h |
| 9.8.12. | South Bridge Base Address 1 Register | Base1 | PCI Config F#1 | | Index 0x14h |
| 9.8.13. | South Bridge Base Address 2 Register | Base2 | PCI Config F#1 | | Index 0x18h |
| 9.8.14. | South Bridge Base Address 3 Register | Base3 | PCI Config F#1 | | Index 0x1Ch |
| 9.8.15. | South Bridge Base Address 4 Register | Base4 | PCI Config F#1 | | Index 0x20h |
| 9.8.16. | South Bridge Primary IDE Timing Register | Prime_IDE_ Time | PCI Config F#1 | | Index 0x40h |
| 9.8.16. | South Bridge Secondary IDE Timing Register | Second_IDE_ Time | PCI Config F#1 | | Index 0x44h |
| 9.8.17. | South Bridge Miscellaneous Register | SB_Misc1 | PCI Config F#1 | | Index 0x48h |
| | | | | | |
| 10.4. | **ISA standard Registers** | | | | |
| 10.4.1. | DMA 1 Controller Registers | DMA_1 | IO | 0000h | |
| 10.4.1. | DMA 1 Channel 0 Base and Current Count | DMA1_CBC0 | IO | 0001h | |
| 10.4.1. | DMA 1 Channel 1 Base and Current Address | DMA1_CBA1 | IO | 0002h | |
| 10.4.1. | DMA 1 Channel 1 Base and Current Count | DMA1_CBC1 | IO | 0003h | |
| 10.4.1. | DMA 1 Channel 2 Base and Current Address | DMA1_CBA2 | IO | 0004h | |

## LIST OF REGISTERS

| Section | Register Name | Mnemonic | Purpose | Address | Access type |
|---|---|---|---|---|---|
| 10.4.1. | DMA 1 Channel 2 Base and Current Count | DMA1_CBC2 | IO | 0005 | |
| 10.4.1. | DMA 1 Channel 3 Base and Current Address | DMA1_CBA3 | IO | 0006h | |
| 10.4.1. | DMA 1 Channel 3 Base and Current Count | DMA1_CBC3 | IO | 0007h | |
| 10.4.1. | DMA 1 Read Status / Write Command Register | DMA1_RSWC | IO | 0008h | |
| 10.4.1. | DMA 1 Request Register | DMA1_RR | IO | 0009h | |
| 10.4.1. | DMA 1 Read Command / Write Single Mask Register | DMA1_RCWSM | IO | 000Ah | |
| 10.4.1. | DMA 1 Mode Register | DMA1_Mode | IO | 000Bh | |
| 10.4.1. | DMA 1 Set / Clear Byte Pointer Flip - Flop | DMA1_SCBPFF | IO | 000Ch | |
| 10.4.1. | DMA 1 Read Temp Register / Master Clear | DMA1_RTMC | IO | 000Dh | |
| 10.4.1. | DMA 1 Clear Mask / Clear All Request | DMA1_CMCAR | IO | 000Eh | |
| 10.4.1. | DMA 1 Read / Write all Mask Register Bits | DMA1_RWMB | IO | 000Fh | |
| 10.4.2. | Interrupt Controller 1 Registers | IC_1 | IO | 0020h | |
| 10.4.3. | Interval Timer Registers | IT_1 | IO | 0040h | |
| 10.4.3. | Interval Timer Register Counter 0 Count | IT_0 | IO | 0040h | |
| 10.4.3. | Interval Timer Register Counter 1 Count | IT_1 | IO | 0041h | |
| 10.4.3. | Interval Timer Register Counter 2 Count | IT_2 | IO | 0042h | |
| 10.4.3. | Command Mode Register | IT_3 | IO | 0043h | |
| 10.4.4. | Port B Register | Port _B | IO | 0061h | |
| 10.4.5. | Port 70 Register | Port_70 | IO | 0070h | |
| 10.4.6. | Interrupt Controller 2 Registers | IC_2 | IO | 00A0h | |
| 10.4.7. | DMA Controller 2 Registers | DMA_Cont2 | IO | | |
| 10.4.7. | DMA 2 Channel 0 Base and Current Address | DMA2_CBA0 | IO | 00C0h | |
| 10.4.7. | DMA2 Channel 0 Base and Current Count | DMA2_CBC0 | IO | 00C2h | |
| 10.4.7. | DMA 2 Channel 1 Base and Current Address | DMA2_CBA1 | IO | 00C4h | |
| 10.4.7. | DMA 2 Channel 1 Base and Current Count | DMA2_CBC1 | IO | 00C6h | |
| 10.4.7. | DMA 2 Channel 2 Base and Current Address | DMA2_CBA2 | IO | 00C8h | |
| 10.4.7. | DMA 2 Channel 2 Base and Current Count | DMA2_CBC2 | IO | 00CAh | |

| Section | Register Name | Mnemonic | Purpose | Address | Access type |
|---|---|---|---|---|---|
| 10.4.7. | DMA 2 Channel 3 Base and Current Address | DMA2_CBA3 | IO | 00CCh | |
| 10.4.7. | DMA 2 Channel 3 Base and Current Count | DMA2_CBC3 | IO | 00CEh | |
| 10.4.7. | DMA 2 Read Status / Write Command Register | DMA2_RSWC | IO | 00D0h | |
| 10.4.7. | DMA 2 Request Register | DMA2_RR | IO | 00D2h | |
| 10.4.7. | DMA 2 Read Command / Write Single Mask Register | DMA2_RCWSM | IO | 00D4h | |
| 10.4.7. | DMA 2 Mode Register | DMA2_Mode | IO | 00D6h | |
| 10.4.7. | DMA 2 Set / Clear Byte Pointer Flip - Flop | DMA2_SCBPFF | IO | 00D8h | |
| 10.4.7. | DMA 2 Read Temporary / Master Clear | DMA2_RTMC | IO | 00DAh | |
| 10.4.7. | DMA 2 Clear Mask / Clear All Requests Register | DMA2_CMCAR | IO | 00DCh | |
| 10.4.7. | DMA 2 Read / Write all Mask Register Bits | DMA2_RWMRB | IO | 00DEh | |
| 10.4.8. | DMA Page Registers | DMA_Page | IO | | |
| 10.4.8. | DMA Page Registers Port 80h | Port_80 | IO | 0080h | (Reserved) |
| 10.4.8. | DMA Page Register Channel 2 | DMA_PRC2 | IO | 0081h | |
| 10.4.8. | DMA Page Register Channel 3 | DMA_PRC3 | IO | 0082h | |
| 10.4.8. | DMA Page Register Channel 1 | DMA_PRC1 | IO | 0082h | |
| 10.4.8. | DMA Page Register Port 84h | Port_84 | IO | 0084h | (Reserved) |
| 10.4.8. | DMA Page Register Port 85h | Port_85 | IO | 0085h | (Reserved) |
| 10.4.8. | DMA Page Register Port 86h | Port_86 | IO | 0086h | (Reserved) |
| 10.4.8. | DMA Page Register Channel 0 | DMA_PRC0 | IO | 0087h | |
| 10.4.8. | DMA Page Register Port 87h | Port_87 | IO | 0088h | (Reserved) |
| 10.4.8. | DMA Page Register Channel 6 | DMA_PRC6 | IO | 0089h | |
| 10.4.8. | DMA Page Register Channel 7 | DMA_PRC7 | IO | 008Ah | |
| 10.4.8. | DMA Page Register Channel 5 | DMA_PRC5 | IO | 008Bh | |
| 10.4.8. | DMA Page Register Port 8Bh | Port_8B | IO | 008Ch | (Reserved) |
| 10.4.8. | DMA Page Register Port 8Ch | Port_8C | IO | 008Dh | (Reserved) |
| 10.4.8. | DMA Page Register Port 8Dh | Port_8D | IO | 008Eh | (Reserved) |
| 10.4.8. | DMA Page Register Port 8Eh | Port_8E | IO | 008Fh | (Reserved) |
| | | | | | |
| 10.5. | **ISA Configuration Registers** | | | 022h/023h | |
| 10.5.1. | Miscellaneous Control Register 0 | Misc_Cont0 | Configuration | | Index 050h |
| 10.5.2. | Miscellaneous Control Register 1 | Misc_Cont1 | Configuration | | Index 051h |
| 10.5.3. | PIRQA Routing control Register 0 | PAR_Cont0 | Configuration | | Index 052h |
| 10.5.3. | PIRQB Routing control Register 0 | PBR_Cont0 | Configuration | | Index 053h |
| 10.5.3. | PIRQC Routing control Register 0 | PCR_Cont0 | Configuration | | Index 054h |

## LIST OF REGISTERS

| Section | Register Name | Mnemonic | Purpose | Address | Access type |
|---|---|---|---|---|---|
| 10.5.3. | PIRQD Routing control Register 0 | PDR_Cont0 | Configuration | | Index 055h |
| 10.5.4. | Interrupt Level Control Register 0 | IRQ_Lev_C_0 | Configuration | | Index 056h |
| 10.5.5. | Interrupt Level Control Register 1 | IRQ_Lev_C_1 | Configuration | | Index 057h |
| 10.5.6. | IPC Configuration Register | IPC_Conf | Configuration | | Index 001h |
| 10.5.7. | ISA Syncronizer Bypass Register | ISA_Sync | Configuration | | Index 059h |
| | | | | | |
| 11. | **IDE Controller** | | Configuration | See IDE Controller Chapter | |
| | | | | | |
| 12.2. | **Local Bus Registers** | | 16 bit access | 22h | |
| 12.3. | **Local Bus Address Decode Registers** | | | 23h | |
| 12.3.1. | I/O Slot Base Register 0 | IOAREG0 | | | Index 0x10h |
| 12.3.1. | I/O Slot Base Register 1 | IOAREG1 | | | Index 0x11h |
| 12.3.1. | I/O Slot Base Register 2 | IOAREG2 | | | Index 0x12h |
| 12.3.1. | I/O Slot Base Register 3 | IOAREG3 | | | Index 0x13h |
| 12.3.5. | I/O Slot Mask Register 0 | IOMREG0 | | | Index 0x14h |
| 12.3.5. | I/O Slot Mask Register 1 | IOMREG1 | | | Index 0x15h |
| | | | | | |
| 12.4. | **Local Bus Timing Registers** | | | | |
| 12.4.1. | Memory Timing Template 0 | TIMEBANK0 | | | Index 0x16h |
| 12.4.2. | Memory Timing Template 1 | TIMEBANK1 | | | Index 0x17h |
| 12.4.3. | I/O Timing Template 0 | TIMEIO0 | | | Index 0x18h |
| 12.4.4. | I/O Timing Template 1 | TIMEIO1 | | | Index 0x19h |
| 12.4.5. | I/O Timing Template 2 | TIMEIO2 | | | Index 0x1Ah |
| 12.4.6. | I/O Timing Template 3 | TIMEIO3 | | | Index 0x1Bh |
| | | | | | |
| 12.5. | **Local Bus Control Register** | | | | |
| 12.5.1. | Control Register | CONTROL | | | Index 0x1Ch |
| 12.5.2. | I/O Width Register | IOWIDTH | | | Index 0x1Eh |
| | | | | | |
| 14.2. | **Power Management Controller Registers:** | | | 022h/023h | |
| 14.2.1. | Timer Register 0 | Timer0 | Configuration | | Index 060h |
| 14.2.2. | Timer Register 1 | Timer1 | Configuration | | Index 061h |
| 14.2.3. | Timer Register 2 | Timer2 | Configuration | | Index 08dh |
| 14.2.4. | System Activity Enable Register 0 | Sys_activ_en0 | Configuration | | Index 062h |
| 14.2.5. | System Activity Enable Register 1 | Sys_activ_en1 | Configuration | | Index 063h |
| 14.2.6. | System Activity Enable Register 2 | Sys_activ_en2 | Configuration | | Index 064h |
| 14.2.7. | House-Keeping Activity Enable Register 0 | HK_activ_en0 | Configuration | | Index 065h |
| 14.2.8. | House-Keeping Activity Enable Register 1 | HK_activ_en1 | Configuration | | Index 066h |

| Section | Register Name | Mnemonic | Purpose | Address | Access type |
|---|---|---|---|---|---|
| 14.2.9. | Peripheral Inactivity Detection Register 0 | Perif_inactiv0 | Configuration | | Index 067h |
| 14.2.10. | Peripheral Activity Detection Register 0 | Perif_activ0 | Configuration | | Index 069h |
| 14.2.11. | Peripheral Activity Detection Register 1 | Perif_activ1 | Configuration | | Index 06Ah |
| 14.2.12. | Address Range 0 Register 0 | Add_rang0-0 | Configuration | | Index 06Bh |
| 14.2.13. | Address Range 0 Register 1 | Add_rang0-1 | Configuration | | Index 06Ch |
| 14.2.14. | SMI Control Register 0 | SMI_cont0 | Configuration | | Index 071h |
| 14.2.15. | SMI Status Register 0 | SMI_stat0 | Configuration | | Index 073h |
| 14.2.16. | SMI Status Register 1 | SMI_stat1 | Configuration | | Index 074h |
| 14.2.17. | Peripheral Inactivity Status Register 0 | Perif_stat0 | Configuration | | Index 075h |
| 14.2.18. | Activity Status Register 0 | Activ_stat0 | Configuration | | Index 077h |
| 14.2.19. | Activity Status Register 1 | Activ_stat1 | Configuration | | Index 078h |
| 14.2.20. | Activity Status Register 2 | Activ_stat2 | Configuration | | Index 079h |
| 14.2.21. | PMU State Register | PMU | Configuration | | Index 07Ah |
| 14.2.22. | General Purpose Register | GP | Configuration | | Index 07Bh |
| 14.2.23. | Clock Control Register 0 | Clock_cont0 | Configuration | | Index 07Ch |
| 14.2.24. | Doze Timer Read Back Register | Doze | Configuration | | Index 088h |
| 14.2.25. | Standby Timer Read Back Register | Standby | Configuration | | Index 089h |
| 14.2.26. | Suspend Timer Read Back Register | Suspend | Configuration | | Index 08Ah |
| 14.2.27. | House-Keeping Timer Read Back Register | HK_timer | Configuration | | Index 08Bh |
| 14.2.28. | Peripheral Timer Read Back Register | Perif_timer | Configuration | | Index 08Ch |

Note 1: X can stand for B (Monochrome Display) or D (Color Display)

Note 2: X is the value of the G_Base and can range from 8 to ...

Note 3: These registers are note described in this publication.

| Register Name | Mnemonic | Purpose | Address | Access type |
|---|---|---|---|---|
| **Configuration Registers** | | | 22h | |
| Configuration Control 1 | CCR1 | IO | 23h | C1h |
| Configuration Control 2 | CCR2 | IO | | C2h |
| Configuration Contol 3 | CCR3 | IO | | C3h |
| SMM Address Region | SMAR | IO | | CDH |
| Device Indentification 0 | DIR0 | IO | | FEh |
| Device Identification 1 | DIR1 | IO | | FFh |

**Table 6-1. CPU Registers located in the ST X86 Manual**

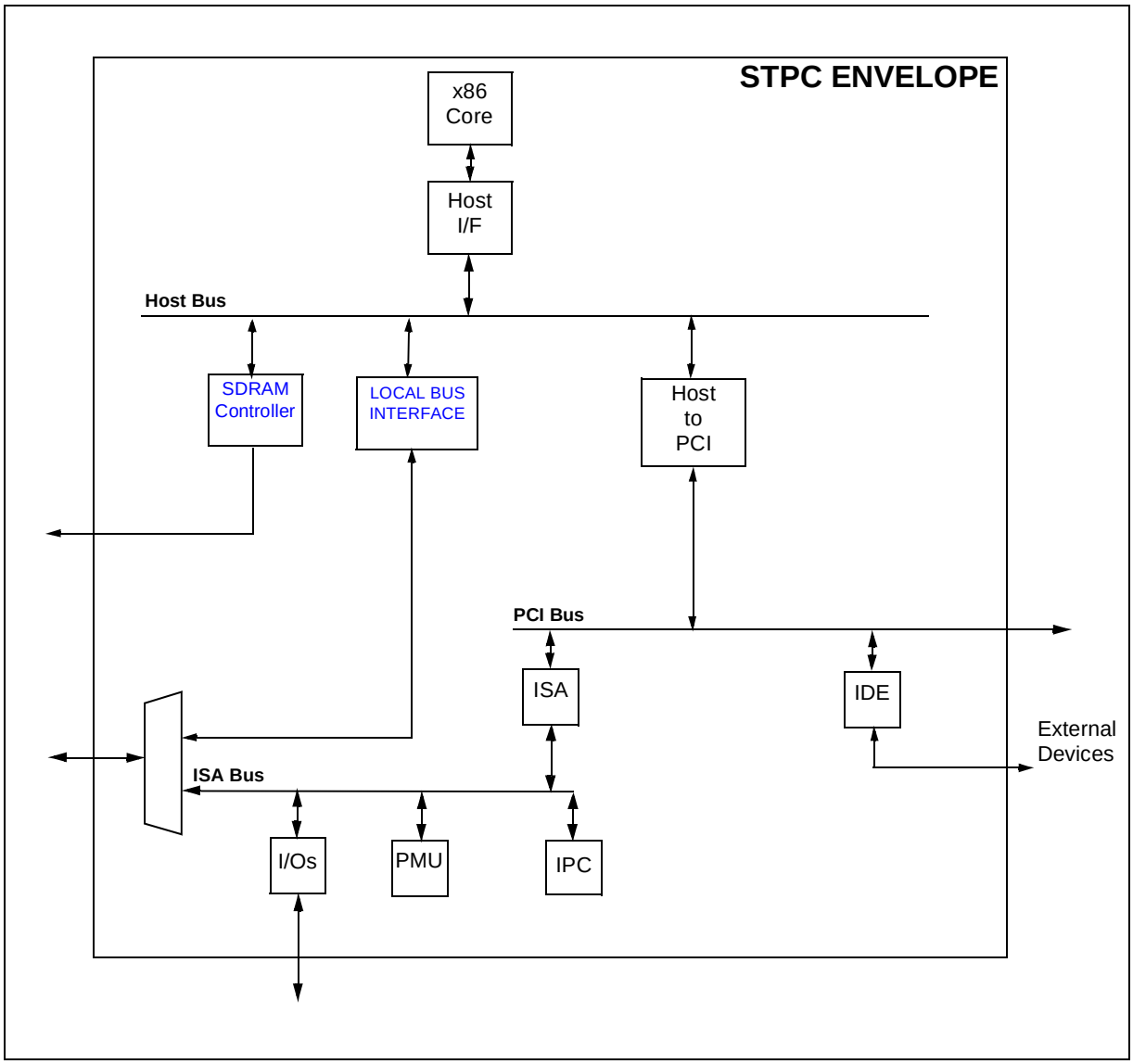
# 7. HOST INTERFACE

## 7.1. INTRODUCTION

This Chapter describes the Memory and I/O Mapping of the STPC with details on how to configure the Cache Memory registers.

The Host is the main interface between the CPU and the other integrated peripherals of the STPC. below illustrates the relation of the integrated devices with reference to the Host Interface.

**Figure 7-1. STPC Host Layout**



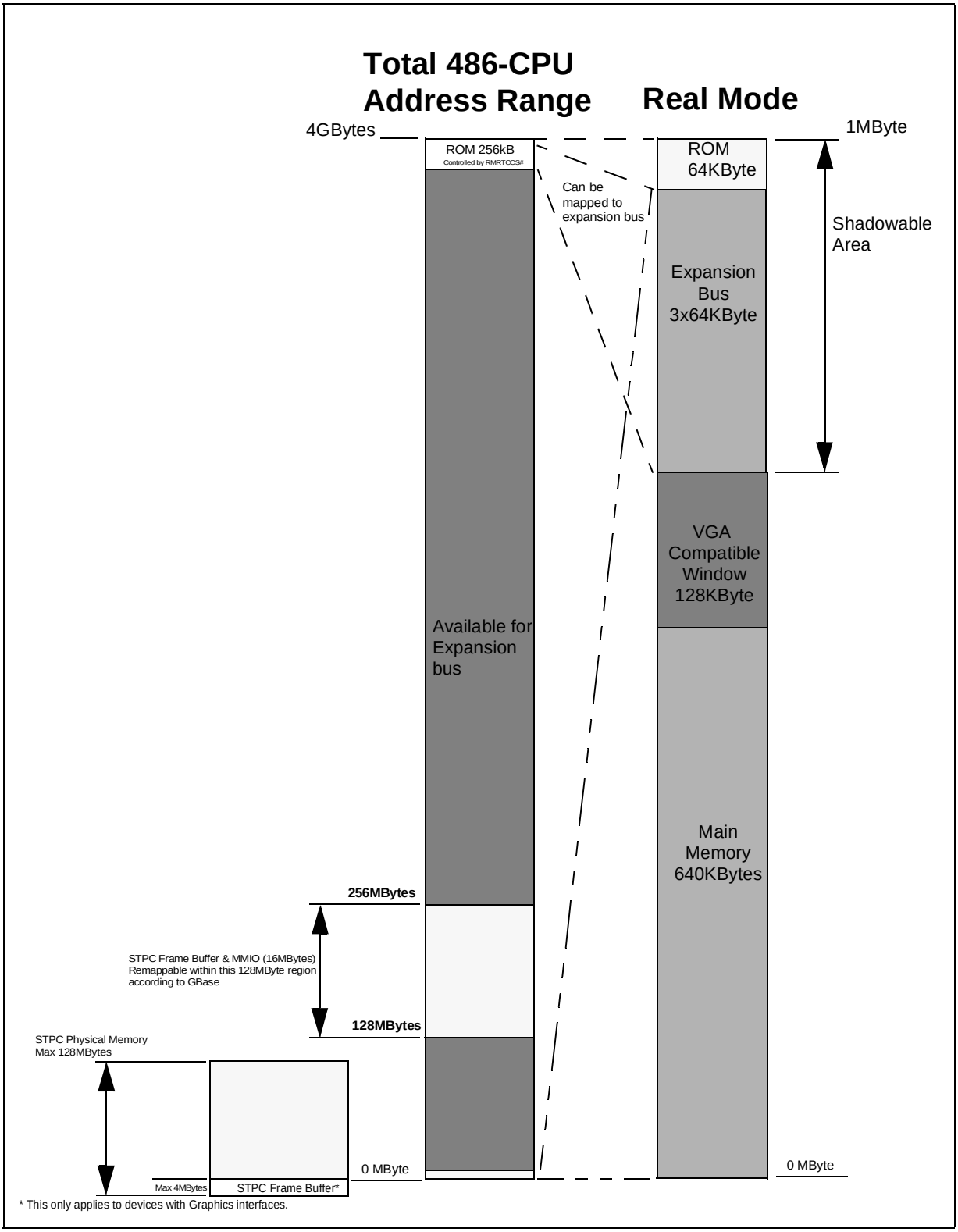



**Figure 7-2. STPC Physical Memory Map**

## 7.2. AGENT DECODING

All agents are decoded on a priority basis for instructions that are sent from the Host onto the host bus. If no agent on the Host Bus claims the cycle, it is then taken by the Host to PCI bridge (PCI North Bridge). If no agent on the PCI bridge claims the cycle it is forarded to the ISA bridge.

For PCI Memory accesses, the cycle is forwarded to the Host Bridge to be decoded in by the SDRAM Controller. PCI Master cycles follow the procedure above. For ISA Master devices, the cycle is first forwarded to the PCI Bridge and follows the procedure described above. ISA Memory cycles are forwarded in the same way as the PCI Memory cycles.

## 7.3. MEMORY ADDRESS MAP

Figure 7-2 illustrate the STPC Memory Map including the general overview of how the SDRAM controller is situated within the complete map including the STPC Frame Buffer Location.

| Memory Region | Address Range | Description |
|---|---|---|
| MAIN MEMORY (640K) | 00000000h 0009FFFFh | Host access maps to the main memory and no ISA or PCI cycle will be initiated. PCI master cycles in this range maps to main memory provided they are not claimed by a PCI Slave. The STPC relies on subtractive decode before initiating an internal memory cycle. ISA master cycles in this range maps to main memory. The STPC will negate IOCHRDY if necessary.<br>The DMA master cycles in this range maps to main memory. The STPC will actively drive the SD bus during target reads and modify main memory for target write transfers.<br>This address segment is considered always cacheable in the L1 cache. PCI and ISA master cycles in this range, require the L1 cache. |
| VGA FRAME BUFFER (128K) | 000A0000h 000BFFFFh | This 128K address segment contains the VGA Frame buffer. Normally this address segment is mapped to the DOS frame buffer located in the main memory. However, if VGA is disabled or the VGA memory map mode is such that the VGA does not occupy the entire 128K address range, the host cycle is forwarded to the PCI bus and if not claimed by a PCI slave, it is further forwarded to the ISA bus.<br>The PCI master cycles in this range, if not claimed by a PCI slave, will be mapped to the main memory or will be forwarded to the ISA bus as per the VGA decode described above.<br>Similarly, the ISA or DMA master cycles will either map to the main memory or will be forwarded to the PCI. If no PCI slave claims the cycle, the STPC assumes existence of an ISA memory device at this address range.<br>This segment is never cacheable. |
| SHADOW (16K) | 000C0000h 000C3FFFh | This 16K address segment can be programmed software to either map to main memory or expansion buses. Further, reads and writes can have different mappings. If mapped to main memory, this segment will behave as the 0-640K segment.<br>If not mapped to main memory, refer to Section 7.2. above<br>If mapped to the main memory, the cacheability of this address range is controlled by software. If mapped to the ISA bus, the ROMCS# signal may optionally be asserted as controlled by software. This allows the system and video/peripheral BIOS to physically reside in a single ROM device. |

| Memory Region | Address Range | Description |
|---|---|---|
| SHADOW (16K) | 000C4000h 000C7FFFh | This range has the same characteristics as that of 000C0000h-000C3FFFh segment, as described above. The shadow control for this address range is provided via Shadow Control register 0 and cacheability and ROM chip-select control via Shadow Control register 3. |
| SHADOW (16K) | 000C8000h 000CBFFFh | This range has the same characteristics as that of 000C0000h-000C3FFFh segment as described above, with the exception of the cacheability attribute. This address range is hardwired to be non-cacheable. Shadow control for this address range is provided via Shadow Control register 0 and ROM chip-select control via Shadow Control register 3. |
| SHADOW (16K) | 000CC000h 000CFFFFh | This range has the same characteristics as that of 000C8000h-000CBFFFh segment, as described above. Shadow control for this address range is provided via Shadow Control register 0 and ROM chip-select control via Shadow Control register 3. This address range is hardwired to be non-cacheable. |
| SHADOW (64K) | 000D0000h 000DFFFFh | This range has the same characteristics as that of 000CC000h-000CFFFFh segment, as described above. Shadow control for this address range is provided via Shadow Control register 1 and can be controlled at 16K resolution. ROM chip-select generation for the entire 64K range can be controlled via Shadow Control register 3. This address range is hardwired to be non-cacheable. |
| SHADOW (64K) | 000E0000h 000EFFFFh | This range has the same characteristics as that of 000CC000h-000CFFFFh segment ,as described above. Shadow control for this address range is provided via Shadow Control register 2 and can be controlled at 16K resolution. ROM chip-select generation for the entire 64K range can be controlled via Shadow Control register 3. This address range is hardwired to be non-cacheable. |
| SHADOW (64K) | 000F0000h 000FFFFFh | This range has the same characteristics as that of 000C0000h-000C7FFFh segment, as described above. Shadow control for this address range is provided via Shadow Control register 3. If not shadowed in the main memory, cycles in this address range which are forwarded to the ISA bus will always results in an ROMCS# assertion. The cacheability of this address segment is controlled via Shadow Control register 3. |
| TOP OF ADDRESSABLE SDRAM MEMORY (1M) | 00100000h | This address segment is mapped to the main memory with the exception of one hole that can optionally be opened in the range 1MBytes to 16 Mytes. The address range defined for the hole is mapped to the expansion buses and is described later in this section. The addressable SDRAM memory can be different from the populated memory due to the memory remapping and the frame buffer. This is described in more detail in a later section.<br>With the exception of the memory holes, this address range has the same characteristics as the 0-640K (compatible DOS memory) range. |
| TOP OF ADDRESSABLE SDRAM MEMORY (4G-256K) | FFFC0000h | All cycles above the addressable SDRAM memory are forwarded to the expansion buses.<br>Host access in this range initiates a PCI cycle and if unclaimed by a PCI slave, they are forwarded to ISA. Note that the ISA address space is only 16M. Higher addresses are aliased to this 16M space. |
| ROM ALIAS (4G-64K) | FFFF0000 FFFFFFFFh | This address segment is an alias of the 64K segment located at F0000h-FFFFFh and has the same attributes except that this segment can never be shadowed into the SDRAM memory.<br>This is also true for address E0000h, D0000h and C0000h provided I/O register Index 51h (see Section 9.13.2. ) is set correctly. |

### *7.3.1. EXTENDED GRAPHICS SEGMENT*

A 16M segment of memory anywhere between Top of addressable SDRAM memory and 256M can be optionally enabled via extended VGA Graphics Registers (GRA). This segment is located at 16M granularity. Refer to the Graphics section for a more detailed description of the layout of this memory segment.

Host access to this region is absorbed by the STPC and are either consumed internally, or initiate a frame buffer memory access.

PCI master access to this region, if not claimed by a PCI slave is absorbed by the STPC and treated the same way as a host access.

This address range by definition is not accessible to ISA and DMA masters, since it must be located at a 16M granularity above the addressable SDRAM memory. The ISA and DMA masters can access only up to 16M address range.

This address segment is always considered non-cacheable.

### *7.3.2. MEMORY HOLE*

The Memory Hole register allows the creation of a hole in the memory space in 1-16M address range. This hole allows mapping expansion bus cards in the AT compatible address range when the addressable main memory size exceeds 16M. A host/PCI/ISA/DMA master cycle in this address range is handled in the same way as a cycle above the addressable memory range described above.

### *7.3.3. SMM MEMORY*

The STPC uses the physical memory behind the CPU address range A0000h - B0000h for the SMM memory. The SMM base address register inside CPU needs to be programmed to A0000h. The initialization of the SMM memory is controlled by RAM System management register and redirects the CPU A0000h-B0000h address range to SMM memory. After the initialization, SMM memory can only be accessed when SMIACT# is active. The cacheability of this segment is hardwired to 0.

### *7.3.4. ADDRESSABLE SDRAM MEMORY*

Addressable SDRAM memory is a function of the size of populated SDRAM, the size of graphic memory, the size of memory hole, and the shadow control of D0000h-DFFFFh and E0000h-EFFFFh segments.

TOPM = The size of total physical SDRAM is defined by SDRAM Bank 3 Register.

TOGM = The size of graphic memory is defined by Graphic memory size register.

MHOLE_SIZE = The size of memory hole defined by Memory Hole Control register.

REMAP_SIZE = 128KB, if none of the 8 x 16KB-segments of D0000h-EFFFFh is enabled for shadow, or 0KB, if any of the 8 x 16KB-segments of D0000h-EFFFFh is enabled for shadow.

The addressable SDRAM memory =

TOPM - TOGM + MHOLE_SIZE + REMAP_SIZE

### *7.3.5. CPU ADDRESS TO SDRAM ADDRESS MAPPING*

The STPC implements a single memory subsystem for both the system as well as the frame buffer memory. In other words, the size of the SDRAM available to the system is reduced by the size of the SDRAM allocated to the frame buffer.

The CPU's concept of a physical address is a logical address to the STPC and is remapped to a SDRAM physical address. This section refers to the CPU's physical address as the "CPU address" and to the SDRAM's physical address as the "SDRAM address".

STPC also defines a memory hole to allow the existence of memory devices on the PCI or ISA buses. The size of the CPU address space is increased by these memory holes, if they exist. CPU address space D0000h to EFFFFh is mapped to the add-in card BIOS area. If this ROM space is not shadowed, then the CPU address space is increased by another 128 KBytes (also see Section 7.6.1. ).

For example:

Total populated SDRAM = 4 MBytes

Frame buffer size = 256 KBytes*

Memory hole size = 1 MByte

Memory hole starting address = 200000h

Shadow feature for D0000h to EFFFFh = disabled

The total CPU memory = 4 MBytes - 256 KBytes + 1 MByte + 128 KBytes = 4 MBytes plus 896 1128KBytes

Since the frame buffer is 256 KBytes*, the system memory is reduced by 256 KBytes and becomes 3 MBytes plus 768 KBytes. Since a 1 MByte memory hole exists, the CPU address space is increased by 1 MByte and becomes 4 MBytes plus 768 KBytes. The CPU address between 3 MBytes plus 768 KBytes and 128 MBytes above this is mapped to the memory hole.

Since the shadowing of the CPU address range D0000h to EFFFFh reserved for add-on card BIOS is not enabled, the CPU memory is increased by 128 KBytes to make use of this SDRAM space that no device accesses. The total CPU memory then becomes 4 MBytes plus 896 KBytes.

* Not applicable to STPC Elite or Vega.

## 7.4. IO ADDRESS MAP

.

**Table 7-1. : IO Map Space**

| IO address | Description | Notes |
|---|---|---|
| 0000h-000Fh | 8237 DMA controller 1 registers. | 1 |
| 0020h-0021h | 8259 Interrupt controller 1 registers. | |
| 0022h | STPC specific configuration registers index port | |
| 0023h | STPC specific configuration registers data port | |
| 0040h-0043h | 8254 Timer/Counter registers. | 1 |
| 0060h-0064h | Keyboard shadow registers. | 1 |
| 0070h-0071h | NMI Mask control registers. | 1 |
| 0080h-008Fh | DMA Page registers. | |
| 0094h | Mother-board VGA enable. | 2 |
| 00A0h-00A1h | 8259 Interrupt controller 2 registers. | 1 |
| 061h | ISA standard Port B. | 1 |
| 00C0h-00DFh | 8237 DMA controller 2 registers. | 1 |
| 0102h | VGA setup register. | |
| 03B4h,03B5h,03BAh | VGA registers. | |
| 03D4h,03D5h,03DAh | | |
| 03C0h-03CFh | | |
| 0CF8h | PCI configuration Address register. | |
| 0CFCh-0CFFh | PCI configuration Data register. | |
| 46E8h | VGA add-in mode enable register. | 2 |

The STPC implements a number of registers in IO address space. This is visible in the registers with access = 0022h/0023h. These regsters use the index and data programming system where the index to which the data is to be written to is programmed in register 0022h and the data is written to register 0023h.

These register occupy the map in the IO space in the table above:Table 7-1

Notes:

1. This address range is partially decoded. Refer to the Register Description section for more details.

2.This address is occupied only if the STPC is strapped to look like a mother-board VGA.

### *7.4.1. PCI CONFIGURATION ADDRESS MAP:*

The STPC occupies Device number 0 slot on the PCI bus and implements a number of registers in PCI configuration address space. These registers occupy the following map (seeTable 7-2) :

**Table 7-2. PCI Configuration Address Space**

| Offset | Description |
|---|---|
| 00h-01h | Vendor Identification register |
| 02h-03h | Device Identification register |
| 04h-05h | PCI Command register |
| 06h-07h | PCI Status register |
| 08h | PCI Revision ID register |
| 40h | PCI Control register |

## 7.5. CACHE RELATED REGISTERS

The STPC supports two caching modes, write-through and write-back. For both modes, sdram read accesses are copied into the cache, and future accesses then return the cache copy with no access to sdram. The situation is different for write access; for the write-through mode, a write updates both the sdram and the cache, whereas in the the write-back mode, a write updates the cache only with the sdram update taking place later. The write-back mode offers improved performance over the write-through mode.

Two cache levels are generally available, level 1 (L1), within the CPU core, and level 2 (L2), within the chipset core (between the CPU and the sdram Controller). For the STPC product range, the L2 cache controller is included but as the required external connection pins are not provided, it is not usable. The L2 cache control registers described below are however used to configure the CPU L1 Cache architecture.

### *7.5.1. CACHE ARCHITECTURE REGISTER 0*

This register controls various attributes of the L2 and L1 cache.

*Cash_Arc0* Access = Regoffset = 0x20h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CPU PAS | BAO | L1 WB | SRAM | | L2 B | L2 WBC | L2 BC |
| Default value after reset = 00000000h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | CPU PAS | **CPU pipelined access.** :<br>0: Not supported<br>1: Supported |
| Bit 6 | BAO | **Burst addressing order.**<br>0: Intel<br>1: Linear |
| Bit 5 | L1 WB | **L1 write back indication.**<br>0: Not supported<br>1: Supported |
| Bit 4-3 | SRAM | **SRAM type.** These bits control the type of SRAMs used to construct L2 cache. (See Table 7-3) |
| Bit 2 | L2 B | **Number of L2 banks.** When programmed to 2 banks, L2 interleaving is enabled.<br>0: One bank<br>1: Two banks |
| Bit 1 | L2 WBC | **L2 write back control.**<br>0: Write through<br>1: Write back |
| Bit 0 | L2 BC | **L2 cache enable.**<br>0: Disabled<br>1: Enabled |

**Table 7-3. Bits 4-3 SRAM Type**

| Bit 4 | Bit 3 | L2 cache SRAM type |
|---|---|---|
| 0 | 0 | asynchronous SRAM |
| 0 | 1 | synchronous burst SRAM |
| 1 | 0 | synchronous burst pipelined SRAM |
| 1 | 1 | reserved |

### *7.5.2. CACHE ARCHITECTURE REGISTER 1*

This register controls various attributes of L2 cache.

*Cash_Arc1* Access = Regoffset = 0x21h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| L2 CS | | | IO NA | S FIFO | | R AWE | Rsv |
| Default value after reset = 00000000h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 7-5 | L2 CS | **L2 cache size.** (See Table 7-4) |
| Bit 4 | IO NA#Enable | **IO NA# Enable.**<br>0: Generate NA#<br>1: Don't generate NA# |
| Bits 3-2 | S FIFO | **Source FIFO low water mark.** These bits control the degree of concurrency between a L1 cache line fill and start of the next memory access. A cache line wide read buffer is implemented.<br>Due to pipelining, it is possible that the buffer may be filled up ahead of drain. Then if the next access is also a read from memory, these bits determine when the next read will be kicked off relative to the drain of the current line from the read buffer. The optimal value is a function of the drain rate of the buffer which depends on the cache RAM type and the programmed burst parameters. A value of '0' for this field is the least optimal value but will always work. (See Table 7-5) |
| Bit 1 | R AWE | **Read around write enable.**<br>0: Reads can not proceed around any posted writes<br>1: Reads can go around a posted write if it is to a different address to the posted writes |
| Bit 0 | Rsv | **Reserved.** |

**Table 7-4. L2 Cache Size**

| Bit 7 | Bit 6 | Bit 5 | L2 Cache Size |
|---|---|---|---|
| 0 | 0 | 0 | 64Kb |
| 0 | 0 | 1 | 128Kb |
| 0 | 1 | 0 | 256Kb |
| 0 | 1 | 1 | 512Kb |
| 1 | 0 | 0 | 1 MB |
| 1 | 0 | 1 | 2 MB |

.

**Table 7-5. Source FIFO Low Water Mark**

| Bit 3 | Bit 2 | Start next read... |
|---|---|---|
| 0 | 0 | only after completely finishing current fill |
| 0 | 1 | when 1 QWORD is still to be emptied |
| 1 | 0 | when 2 QWORDs are still to be emptied |
| 1 | 1 | when 3 QWORDs are still to be emptied |

### 7.5.3. CACHE ARCHITECTURE REGISTER 2

*Cash_Arc2* Access = Regoffset = 0x22h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | SHDD | CWEPW | CDHAWE | BAWS | | TAWS | |
| Default value after reset = 11111111h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | Rsv | **Reserved.** |
| Bit 6 | SHDD | **Slow host data driver.**<br>0: Fast, One clock to drive the HD bus<br>1: Slow, two clocks to drive HD bus |
| Bit 5 | CWEPW | **Cache write enable pulse width.**<br>0: 1 clock wide<br>1: 1.5 clocks wide<br>Applicable to asynchronous SRAMs only. Must be '0' for synchronous SRAMs. |
| Bit 4 | CDHAWE | **Cache data hold after write enable.**<br>0: Data removed in the same clock as write enable trailing edge<br>1: Data is kept valid for 1 extra clock after write enable<br>Must be a '1' if 1.5 clocks wide write enable pulse width is selected via bit 5 above. |
| Bits 3-2 | BAWS | **Burst access wait states.** (See Table 7-6) |
| Bits 1-0 | TAWS | **Tag access wait states.** (See Table 7-7) |

**Table 7-6. Burst Access Wait States**

| Bit 3 | Bit 2 | Burst access wait states |
|---|---|---|
| 0 | 0 | fastest |
| 0 | 1 | 1 clock slower than fastest |
| 1 | 0 | 2 clocks slower than fastest |
| 1 | 1 | 3 clocks slower than fastest |

**Table 7-7. Tag Access Wait States**

| Bit 1 | Bit 0 | Tag access wait states |
|---|---|---|
| 0 | 0 | fastest |
| 0 | 1 | 1 clock slower than fastest |
| 1 | 0 | 2 clocks slower than fastest |
| 1 | 1 | 3 clocks slower than fastest |

## 7.6. ADDRESS DECODE RELATED REGISTERS

The following registers are all 8-bit. They are accessed by setting the Configuration Index Port (22h) to the Configuration Index (C.I.) shown, and then reading or writing the appropriate values from the Configuration Register Data Port (23h).

### *7.6.1. MEMORY HOLE CONTROL REGISTER*

This 8-bit register defines the enable, size, and starting address of memory hole. Any memory accesses to this memory hole are directed to PCI/ISA bus.

*MEM_HOLE* Access = Regoffset = 0x24h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MHE | MHS | | | MHSA | | | |
| Default value after reset = 00000000h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | MHE | **Memory Hole Enable.** This bit controls the enable of memory hole function.<br>0 = disabled<br>1 = enabled |
| Bits 6-4 | MHS | **Memory Hole Size.** These bits control the size of memory hole (See Table 7-8) |
| Bits 3-0 | MHSA | **Memory Hole Start Address.** These bits control the bits 23-20 of the memory hole starting address. The memory hole starting address must be aligned to the hole size. |

**Table 7-8. Memory Hole Size**

| Bit 6 | Bit 5 | Bit 4 | Memory Hole Size |
|---|---|---|---|
| 0 | 0 | 0 | 1 MB |
| 0 | 0 | 1 | 2 MB |
| 0 | 1 | 1 | 4 MB |
| 1 | 1 | 1 | 8 MB |
| others | | | reserved |

Programming notes:

This memory hole is also non-cacheable.

### *7.6.2. SHADOW CONTROL REGISTER 0*

This 8-bit register controls the read/write attributes of the memory located at C0000h-CFFFFh. Each 16k of the whole 64k is controlled by 2 bits, one for read and one for write.

*SHADOW_0* Access = Regoffset = 0x25h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RC1 | WC1 | RC2 | WC2 | RC3 | WC3 | RC4 | WC4 |
| Default value after reset = 00000000h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | RC1 | **Read Control CC000h-CFFFFh.** This bit controls the read attribute of the CC000h-CFFFFh memory.<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 6 | WC1 | **Write Control CC000h-CFFFFh.** This bit controls the write attribute of the CC000h-CFFFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |
| Bit 5 | RC2 | **Read Control C8000h-CBFFFh.** This bit controls the read attribute of the C8000h-CBFFFh memory.<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 4 | WC2 | **Write Control C8000h-CBFFFFh.** This bit controls the write attribute of the C8000h-CBFFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |
| Bit 3 | RC3 | **Read Control C4000h-C7FFFh.** This bit controls the read attribute of the C4000h-C7FFFh memory.<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 2 | WC3 | **Write Control C4000h-C7FFFFh.** This bit controls the write attribute of the C4000h-C7FFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 1 | RC4 | **Read Control C0000h-C3FFFh.** This bit controls the read attribute of the C0000h-C3FFFh memory.<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 0 | WC4 | **Bit 0 Write Control C0000h-C3FFFFh.** This bit controls the write attribute of the C0000h-C3FFFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |

Programming Notes:

There is single cacheability bit for the 32k Video BIOS segment (C0000h-C7FFFh) located in Shadow Control register 2. C7FFFFh-CFFFFh segment has the cacheability bit hardwired to ‘1’ (enabled). If shadow is enabled for read/write cycles, read from and write to this area are directed to the system memory. Or else the cycles are forwarded to the expansion buses.

### *7.6.3. SHADOW CONTROL REGISTER 1*

Similar to Shadow Control Register 0, this 8-bit register controls the read/write attributes of the memory located at D0000h-DFFFFh.

*SHADOW_1* Access = Regoffset = 0x26h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SRC | SWC | SWC | SWC | SRC | SWC | SRC | SWC |
| Default value after reset = 00000000h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | SRC | **Shadow Read Control DC000h-DFFFFh.** This bit controls the read attribute of the DC000h-DFFFFh memory.<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 6 | SWC | **Shadow Write Control DC000h-DFFFFh.** This bit controls the write attribute of the DC000h-DFFFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |
| Bit 5 | SWC | **Shadow Write Control D8000h-DBFFFh.** This bit controls the read attribute of the D8000h-DBFFFh memory.<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 4 | SWC | **Shadow Write Control D8000h-DBFFFh.** This bit controls the write attribute of the D8000h-DBFFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |
| Bit 3 | SRC | **Shadow Read Control D4000h-D7FFFh.** This bit controls the read attribute of the D4000h-D7FFFh memory<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 2 | SWC | **Shadow Write Control D4000h-D7FFFh.** This bit controls the write attribute of the D4000h-D7FFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |
| Bit 1 | SRC | **Shadow Read Control D0000h-D3FFFh.** This bit controls the read attribute of the D0000h-D3FFFh memory.<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 0 | SWC | **Shadow Write Control D0000h-DFFFFh.** This bit controls the write attribute of the D0000h-DFFFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |

Programming Notes: This entire 64K segment has the cacheability bit hardwired to ‘0’ (disabled)

### *7.6.4. SHADOW CONTROL REGISTER 2*

Similar to Shadow Control Register 0, this 8-bit register controls the read/write attributes of the memory located at E0000h-EFFFFh.

*SHADOW_2* Access = Regoffset = 0x27h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RC | WC | RC | WC | RC | WC | RC | WC |
| Default value after reset = 00000000h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | RC | **Read Control EC000h-EFFFFh.** This bit controls the read attribute of the EC000h-EFFFFh memory.<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 6 | WC | **Write Control EC000h-EFFFFh.** This bit controls the write attribute of the EC000h-EFFFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |
| Bit 5 | RC | **Read Control E8000h-EBFFFh.** This bit controls the read attribute of the E8000h-EBFFFh memory.<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 4 | WC | **Write Control E8000h-EBFFFFh.** This bit controls the write attribute of the E8000h-EBFFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |
| Bit 3 | RC | **Read Control E4000h-E7FFFh.** This bit controls the read attribute of the E4000h-E7FFFh memory.<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 2 | WC | **Write Control E4000h-E7FFFFh.** This bit controls the write attribute of the E4000h-E7FFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |
| Bit 1 | RC | **Read Control E0000h-E3FFFh.** This bit controls the read attribute of the E0000h-E3FFFh memory.<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 0 | WC | **Bit 0 Write Control E0000h-EFFFFh.** This bit controls the write attribute of the E0000h-EFFFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |

Programming Notes: This entire 64K segment has the cacheability bit hardwired to ‘0’ (disabled)

### *7.6.5. SHADOW CONTROL REGISTER 3*

This 8-bit register controls the cacheability attributes of C0000h-C7FFFh and F0000h-FFFFFh shadow segments.

*SHADOW_3* Access = Regoffset = 0x28h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SMRAM | CCF | CCC | Rsv | | | RCF | WCF |
| Default value after reset = 00000000h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | SMRAM | **SDRAM Initialization Enable.** This bit controls whether CPU accesses in A0000h-BFFFFh address range are decoded as VGA frame buffer access or SMRAM access.<br>0 = A0000h-BFFFFh is interpreted as VGA frame buffer access<br>1= A0000h-BFFFFh is interpreted as SMRAM access.<br>The STPC allows for 128KBytes of SMRAM. Physically this memory is located in the system memory behind the higher address range. This area of the system memory is normally unused since this address range is normally mapped to frame buffer which has its own memory.<br>When the CPU is operating in SMM, accesses in the range of A0000-BFFFFh goes to SMRAM instead of VGA frame buffer. The rest of the address map remains unchanged.<br>The address range A0000h-BFFFFh is always non-cacheable. |
| Bit 6 | CCF | **Cache Control F0000h-FFFFFh.** This bit controls the cacheability of F0000h-FFFFFh block when the shadow function is enabled.<br>0 = cacheability disabled<br>1 = cacheability enabled |
| Bit 5 | CCC | **Cache Control C0000h-C7FFFh.** This bit controls the cacheability of C0000h-C7FFFh block when the shadow function is enabled.<br>0 = cacheability disabled<br>1 = cacheability enabled |
| Bits 4-2 | Rsv | **Reserved.** |
| Bit 1 | RCF | **Read Control F0000h-FFFFFh.** This bit controls the read attribute of F0000h-FFFFFh memory.<br>0 = shadow disabled for read cycle<br>1 = shadow enabled for read cycle |
| Bit 0 | WCF | **Bit 0 Write Control F0000h-FFFFFh.** This bit controls the write attribute of F0000h-FFFFFh memory.<br>0 = shadow disabled for write cycle<br>1 = shadow enabled for write cycle |

Programming notes:

The rest of the shadow RAM segments have the cacheability bits hardwired to ‘0’ (disabled). This register also provides control over the address range for which ROM chip-select (ROMCS#) will be asserted allowing various BIOSes (system, video, disk etc.) to be implemented in a single part. Bit 7 of this register also provides accessibility to the SMM mode RAM (SMRAM).

## 7.7. HOST SDRAM CONTROLLER REGISTERS

The STPC manages 4 Memory Banks (if DIMM sockets are used they can be populated with either single or double sided 64-bit data DIMMs). For SDRAM densities are supported see the datasheet Section 6.3.3. .

Configuration registers 30-33 provide the top addresses for each bank. Any bank can be skipped by the top addresses of two consecutive banks having the same address.

### *7.7.1. MEMORY BANK 0 REGISTER - C.I. 30H (MEMORY__BANK0)*

This 8-bit register controls the top address of memory bank 0. Register bit 7-0 corresponding to memory address bits 27-20.

Bank 0 Top Address = Memory Bank0 size in MBytes -1.

Bank 1 Top Address = Memory Bank0 + Memory Bank1 size in MBytes -1

This register defaults to 07h.

Example 1:

Memory Bank0 = 4MB
Memory Bank1 = 4MB

Bank 0 Top Address = 4 -1= 3= 03h

Bank 1 Top Address = 4 + 4 - 1 = 07h

Bank 2, 3 Top Address = 07h

Example 2: for use with double sided DIMMs

Memory Bank0 = 32MBytes (dbl. sided DIMMS)
Memory Bank1 = 32MBytes (dbl. sided DIMMS)

Bank 0 Top Address = 16 - 1 = 15 = 0Fh

Bank 1 Top Address = 16 + 16 - 1 = 31 = 1Fh

Bank 2 Top Address = 32 + 16 - 1 = 47 = 2Fh

Bank 3 Top address = 48 + 16 - 1 = 63 = 3Fh

### *7.7.2. MEORY BANK 1 REGISTER - C.I. 31H (MEMORY_BANK1)*

This register controls the top address of memory bank 1.

### *7.7.3. MEMORY BANK 2 REGISTER - C.I. 32H (MEMORY_BANK2)*

This register controls the top address of memory bank 2.

### *7.7.4. MEMORY BANK 3 REGISTER - C.I. 33H (MEMORY_BANK3)*

This register controls the top address of memory bank 3.

### *7.7.5. SDRAM REFRESH REGISTER*

This refresh register also contains a number of host clock settings for the SDRAM refresh interval.

*SDRAM_Ref* Access = 0022h/0023h Regoffset = 039h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RE | RC | | | | | | |
| Default value after reset = 30h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | RE | **Refresh Enable.** This bit must be programmed to '0' for normal operation |
| Bits 6-0 | RC | Refresh Cycle. (HCLK frequency in MHz * 15.6us ) >> 4 |

* Examples: (rounded down to nearest integer)

round_down( (75MHz * 15.6us ) >> 4 ) = 73 = 49h

round_down( (66MHz * 15.6us ) >> 4 ) = 65 = 41h

round_down( (60MHz * 15.6us ) >> 4 ) = 58 = 3Ah

round_down( (50MHz * 15.6us ) >> 4 ) = 48 = 30h

Programming notes:

The refresh interval should be reset to the smallest likely run time value (typically 48 HCLKs) to provide warm up cycles for the SDRAM.

A refresh request is generated whenever this register is written to without setting the refresh enable bit.

### *7.7.6. PRESENTS DETECT REGISTER - C.I. 97H*

This register is read through the DDC register in Section 8.3.4. .

### 7.8. ACCESSING CONFIGURATION REGISTERS

The Host interface and the Local Bus Interface are programmed identically. To access all the internal configuration registers, the programmer will need to program the Index (address) and data registers of the the required interface through port 22h/23h. The principle of the programming is to fix the address of device to a location that the user requires and to always adress it at that location. The steps required to access any of the internal registers are as follows:

1. Select the interface you want to programme. Each interface is described as a device and both have a number. The Local Bus device number is 6 and the Host device number is 7. The below example code describes how the devices are accessed and the Host device is used.

2. Select Host interface base programming option in RBI, by writing 0x00 and 0x07 (0x06 for the Local Bus, see Section 11.4. for more details) in register 0x11 (index value 0x11 accessed through I/O 22h/23h address) and 0x10 respectively.

```
IOWRITE8(0x22,0x10);

IOWRITE8(0x23,0x07);

IOWRITE8(0x22,0x11);

IOWRITE8(0x23,0x00);
```

3. Select the Host interface address. Assume that HOST_BASE is the address of the Host interface I/O space.

```
IOWRITE8(0x22,0x12);

IOWRITE8(0x23,(HOST_BASE &0xFF) | 0x03);

IOWRITE8(0x22,0x13);

IOWRITE8(0x23,HOST_BASE >>8);
```

The host interface registers are then accessed with HOST_BASE as the index register and HOST_BASE+4 as the data register, as shown below.

4. Writing into any Internal Register of the Host Interface:

```
IOWRITE8(HOST_BASE,offset);

IOWRITE32(HOST_BASE+4,data);
```

Here the “offset” index address is as mentioned in the register table shown above. The 32-bit “data” is written into the register.

5. Reading from any internal register of Host Input

```
IOWRITE8(HOST_BASE,offset);

IOREAD32(HOST_BASE+4,data);.
```

Here the “offset” index address is as mentioned in the register tables shown above. The 32-bit “data” is expected to be read from the internal register.

One constraint is that the Local Bus address must be set at multiples of 8h.

# 8. SDRAM CONTROLLER

## 8.1. INTRODUCTION

This chapter describes the mapping of the CPU memory and IO address spaces.

The STPC uses a Unified Memory Architecture; the system memory and the graphics buffers use the same memory space. This chapter provides information on the memory address map and the graphics memory usage, together with information on the arbitration logic which resolves accesses to the main memory. Details of memory shadowing and cachability by software control and the Memory Hole for ISA BIOS are also given. The actual interface to the external SDRAM modules is presented. Also introduced in this chapter are the PCI configuration space mapping registers, further details are in the chapter relating to the PCI Bus Controller.

## 8.2. MEMORY CONTROLLER

The STPC handles the memory data bus directly, controlling from 8 MBytes to 128 MBytes. The SDRAM controller supports accesses to the Memory Banks to/from the CPU (via the host) and the local Bus which can be populated with either single-sided or double-sided 72-bit (4-bit parity) memory devices. Parity is not supported.

The SDRAM controller only supports 64 bit wide Memory Banks.

The SDRAM Controller supports buffered or unbuffered SDRAM but not EDO or FPM modes. SDRAMs must support Full Page Mode Type access.

The STPC Memory Controller provides various programmable SDRAM parameters to allow the SDRAM interface to be optimised for different processor bus speeds SDRAM speed grades and CAS Latency.

## 8.3. SDRAM REGISTER ACCESS

These registers are used to configure the SDRAM controller.

### *8.3.1. REGISTER 0*

This is a 31-bit configuration register for the SDRAM controller block:

*MEM_REG0* Access = Regoffset = 84C6000h

| | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RCASLCY | | | REGD | MRSR | RASo | LHDI | Rsv | Rsv | CASLat | | | Config | | PRA |
| Default value after reset = 31x32198376h | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRA | Rsv | | | | RRW | | | BT | BL | | | RCT | | | |
| Default value after reset = 31x32198376h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 30-28 | RCASLCY | **Read CAS Latency (RCASLCY)** should be equal to CASLCY but is left programmable for debug purpose. |
| Bit 27 | REGD | **Registered DIMM**, Indicate if we use registered DIMMs ‘1’ or not ‘0’. |
| Bit 26 | MRSR | **Mode Register Set Request,** If set to1, we update the SDRAM chips corresponds to the value programmed Bits [16:0]. |
| Bit 25 | RASo | **RAS on/off**, When we finish a read or a write, if set to 1 go into RACTIVE, if set to 0, go into PRECHARGE and the IDLE. |
| Bit 24 | LHDI | **Latch_Host_Data_In** for host input data. If set to 0, select MD[63:0] directly from SDRAM, if set to 1, select lactched MD[63:0] by RDCLK (see bits 3-0 in Register 1). |
| Bit 23 | Rsv | Reseved. |
| Bit 22 | Rsv | Reserved |
| Bits 21-19 | CASLat | **CAS Latency** This is abbreviated as CL in SDRAM datasheets and is defined as the number of clock cycles for the data held for after CAS goes low. |
| Bits 18-17 | Config | **CONFIGURATION** of a DIMM. (See Table 8-1)<br>These bits are used to determine the maximum burst length (full page burst).<br>If the DIMMS are populated with a different kind of memory, the 2 bits are programmed to maximise the burst length using the minimum amount of the memory. |
| Bits 16-15 | PRA | **Precharge to Row Active** number of cycles (called tRP in datasheets). |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 14-11 | Rsv | **Reserved.** Should be set to ‘0000’. |
| Bits 10-8 | RRW | **RACTIVE to Read/Write**(Called tRCD in datasheets). |
| Bit 7 | BT | **Burst Type**.Should be set to ‘0’. |
| Bits 6-4 | BL | **Burst Length**. Should be set to ‘111’ |
| Bits 3-0 | RCT | **Refresh Cycle Timing** (Called tRC in SDRAM datasheets). |

**Table 8-1. Memory Bank Configuration**

| Bit 18 | Bit 17 | Memory Bank Configuration | Maximum Burst Length |
|---|---|---|---|
| 0 | 0 | [4Mx4]x16 | 1024 |
| 0 | 1 | [2Mx8]x8 | 512 |
| 1 | 0 | [1Mx16]x4 | 256 |
| 1 | 1 | Reserved | |

### *8.3.2. REGISTER 1*

This 6-bit register is used for the read clock scheme. See Chapter 6.3 “Clock considerations” for more details. This delay can be set up to 3.5 ns beyond the 15ns required from the previous MCLKI edge.

*MEM_REG1* Access = Regoffset = 84C6004h

| | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| | CSMEM | MEM | RCDP | | | |
| Default value after reset = 000000h | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 5 | CSMEM | **CS_MEM16_OE** It is programmed as bit 4 in case of 64Mbits and 128Mbits and is programmed to 1 otherwise. |
| Bit 4 | MEM | **MEM16_OE**_, This bit is set to ‘1’ to get 16mA output enabled, set to ‘0’ to get 8mA output enabled |
| Bits 3-0 | RCDP | **Read Clock Delay Programmation,** 0000 for smallest delay to 1111 for largest delay |

### *8.3.3. REGISTER 2 (MEM_REG2) 84C6008H*

This 2-bit register is used to determine the type of SDRAM in use.

| *MEM_REG2* | Access = | Regoffset = 84C6008h |
|---|---|---|

| | 1 | 0 |
|---|---|---|
| | SDRAM | |
| Default value after reset = 000000h | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 1-0 | SDRAM | **SDRAM type.** (See Table 8-2) |

**Table 8-2. SDRAM Type**

| Bit 1 | Bit 0 | Description |
|---|---|---|
| 0 | 0 | 16 Mbits SDRAMs |
| 0 | 1 | 64 Mbits or 128 Mbits 2 SDRAM banks |
| 1 | 0 | 64 Mbits or 128 Mbits 4 SDRAM banks |

### *8.3.4. DDC CONTROL REGISTER DDCR*

This register is used for Presence detect.

DDCR Access = 022h/023h Regoffset = 0x95h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DDCWD | | DDCRD | | Rsv | | | Rsv |
| Default value after reset = 11111111h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-6** | **DDCWD** | **DDC Write Data.** These two bits drive the DDC[1:0] open collector outputs. Writes to these bits affect the DDC[1:0] pins.<br>The DDC[1:0] pins are open collector outputs which are externally pulled up. Thus, programming either of these bits to a one disables the output driver and allows the pin to act as an input whose status can be read via bits 5-4 of this register.<br>Note that reads from these bits return the value of data last written to this register. This may not be the same as the data actually on the bus if another master is driving it. Bits 5-4 of this register accurately reflect the data on the bus no matter who is driving it. |
| **Bits 5-4** | **DDCRD** | **DDC Read Data.** These read-only bits return the read status of the DDC[1:0] pins. |
| **Bits 3-1** | **Rsv** | **Reserved.** These bits read are both readable and writable and must be programmed to ones to ensure future compatibility. |
| **Bits 0** | **Rsv** | **Reserved.** This bit must be programmed to ‘1’ for correct operation. |

## 8.4. MEMORY CLOCK REGISTERS

The MCLK register is used for Memory Clock control operations.

### *8.4.1. MCLK CONTROL REGISTER 0*

*MCLK00* Access = Regoffset = 0x40h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Uns | 4-bitDIV | | | | 8-bitP | | |
| Default value after reset = 0x5B | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | Uns | Unused. |
| Bits 6- 3 | 4-bitDIV | This is the 4-bit M (divisor) value of the Memory synthesiser. |
| Bits 2-0 | 8-bitN | This is the 3-bit P (exponent) value of the Memory clock synthesizer. |

This register defaults to 0x5B at reset. This value when combined with the default value of the other half of this pair results in a memory clock of 80.05 MHz assuming 14.318 MHz oscillator clock as the reference input.

For MCLK frequency programming values see table below:Table 8-3.

**Note:** If programming MCLK to a frequency not equal to HCLK, strap MD[5] must be disabled (i.e. set to 0) to remove the synchronization between these two clock signals (see Atlas Datasheet, STRAP OPTION section for further details).

$$MCLK = \frac{2 \times 14.31818 \times N}{M \times 2^{P}}$$

Constraints: $1 \leqslant M \leqslant 255$

$1 \leqslant N \leqslant 255$

$0 \leqslant P \leqslant 5$

$$1\ MHz \leqslant \frac{14.31818}{M} \leqslant 2\ MHz$$

$$200\ MHz \leqslant \frac{2 \times 14.31818 \times N}{M} \leqslant 622\ MHz$$

### *8.4.2. MCLK CONTROL REGISTER 1*

*MCLK01* Access = Regoffset = 0x41h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 8-bitN | | | | | | | |
| Default value after reset = 0xEC | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 7- 0 | 8-bitN | These are bits 4-0 of the 8-bit N (multiplier) value of the Memory clock synthesiser. |

This register defaults to 0xEC at reset. This value when combined with the default value of the other half of this pair results in a memory clock of 80.05 MHz assuming 14.318 MHz oscillator clock as the reference input.

For MCLK frequency programming values see Table 8-3

**Table 8-3. MCLK Control Register Address 22 Index 40h, 41h**

| MHz | Reg1,Index 41h | Reg0, Index 40h | Actual Freq. | m | n | p |
|---|---|---|---|---|---|---|
| 100 | 9Ah | 5Ah | 100.227260 | 11 | 9Ah | 2 |
| 95 | 92 | 5Ah | 95.020649 | 11 | 146 | 2 |
| 90 | B0h | 72h | 89.999989 | 14 | 176 | 2 |
| 85 | 5Fh | 42h | 85.014194 | 8 | 95 | 2 |
| 80 | 7Bh | 5Ah | 80.051643 | 11 | 123 | 2 |
| 75 | 88h | 6Ah | 74.895095 | 13 | 136 | 2 |
| 66 | 95h | 43h | 65.965901 | 8 | 149 | 3 |
| 60 | 6Dh | 6Ah | 60.026216 | 13 | 109 | 2 |
| 55 | A9h | 5Bh | 54.994828 | 11 | 169 | 3 |
| 50 | 4Dh | 5Ah | 50.113630 | 11 | 77 | 2 |
| 45 | B0h | 73h | 44.999994 | 14 | 176 | 3 |
| 8 | 3Fh | 3Dh | 8.053978 | 7 | 63 | 5 |

# 9. PCI CONTROLLERS

## 9.1. INTRODUCTION

The PCI bus is the main data communication link to the STPC chip. Two PCI devices are present internally in the STPC, a “North Bridge” and a “South Bridge”. The STPC also contains a PCI arbiter which arbitrates between the two bridges and for up to three external PCI devices. Figure 9-1 below shows the layout of the PCI controllers within the STPC. Please refer to *PCI Specification 2.1*, from PCI-SIG, for further details of the PCI bus standard.

The *North Bridge* translates the appropriate host bus I/O and Memory cycles to the PCI bus. It also supports generation of Configuration cycles on the PCI bus. The Configuration Address register allows for the remapping of host CPU I/O cycles, in the address range 0xCF8h-0xCFFh, to configuration cycles on the PCI bus.

The North Bridge, as a PCI bus agent (host bridge class), fully complies with PCI specification 2.1. The North Bridge also implements the PCI mandatory header registers in Type 0 PCI configuration space for easy porting of PCI-aware system BIOS. The North Bridge is assigned the Device Number 0xBh, which corresponds to the IDSEL on AD11 signal. PCI configuration registers of the North Bridge are accessible by the Type 0 PCI configuration cycles generated at Device number 0xBh.

*The South Bridge* controller responds to PCI configuration read and write transactions. The South Bridge is assigned the Device Number 0xCh, which corresponds to the IDSEL on AD12 signal. PCI configuration registers of the South Bridge are accessible by the Type 0 PCI configuration cycles generated by the North Bridge.

The South Bridge, as a PCI bus agent (expansion bridge class), fully complies with PCI specification 2.1. The South bridge implements two PCI functions:

Function 0, PCI to ISA bridge,
Function 1, IDE controller.

As per the PCI specification, the South Bridge will respond to both function 0 and function 1 configuration cycles directed to configuration slot 12 (AD12).

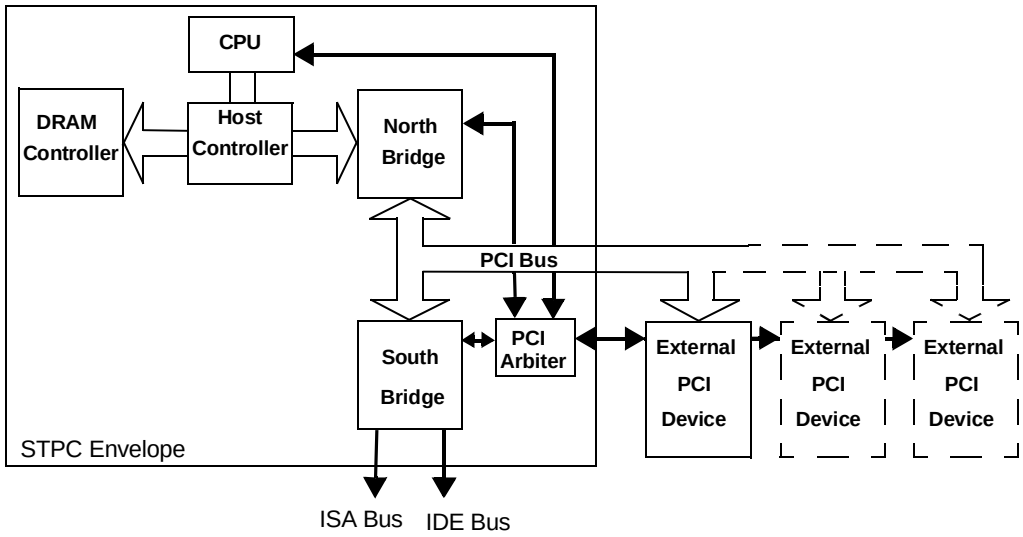



**Figure 9-1. PCI Layout**

### *9.1.1. PCI ADDRESS DECODE*

The only positive decode carried out is for the IDE controller PIO registers. The decode address ranges are dictated by the IDE configuration registers, see IDE section for details. ISA resources are accessed only via subtractive decode.

### *9.1.2. PCI ERROR HANDLING*

Under control of South Bridge configuration registers, one or more of the following events can generate a 1 PCICLK long pulse on SERR#, which in turn can be made to generate an NMI to the CPU.

ISA initiated transaction ending in target abort.

### *9.1.3. PCI ARBITER*

The PCI arbiter controls access to the PCI bus when several bus masters are present in the system. Whenever a further potential bus master needs to gain access to the bus, it asserts its request. The arbiter then asserts a system hold condition, which eventually causes a hold signal to be asserted to the CPU. The CPU finishes the current instruction, tristates the internal bus and asserts a hold acknowledge. This eventually causes the assertion of a system hold acknowledge. Once the system hold acknowledge is asserted, the arbiter asserts a grant to whichever requesting master is in the front of the line in a round-robin chain. When there are no requests pending or when the CPU is requesting the bus and it is in the front of the line, control of the bus is passed back to the CPU by the negation of the system hold condition.

## 9.2. ACCESSING THE PCI CONFIGURATION REGISTERS

The PCI configuration registers are accessed, from the CPU, using two 32-bit registers, mapped as I/O at CF8h and CFCh.

Each read from and write to the PCI configuration registers must be done by:

Writing the 32-bit address of the PCI config. register using type 0 format at I/O CF8h.

Reading or Writing 32-bit data at CFCh

All PCI configuration registers, inside the North and South bridges and all other external PCI devices, are seen from the CPU through those 2 x 32-bit registers.

An illustration of these registers is shown in Table 9-1 & Table 9-2.

**Table 9-1. Register CF8h**

| 31 | 30 ----- 24 | 23 ---------- 16 | 15 ---------- 11 | 10 ---------- 8 | 7 ---------- 2 | 1 0 |
|---|---|---|---|---|---|---|
| Enable | Reserved | Bus number | Device number | Function number | Register number | 0 |

**Table 9-2. Register CFCh**

| 31 ---------- 24 | 23 ---------- 16 | 15 ---------- 8 | 7 ---------- 0 |
|---|---|---|---|
| Byte 3 | Byte 2 | Byte 1 | Byte 0 |

### 9.3. CONFIGURATION ADDRESS REGISTER

This is a 32-bit register accessible only via double-word IO read and write cycles.

*Config_Address* Access = 0xCF8h Regoffset =

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PCI | Rsv | | | | | | | BN | | | | | | | |
| Default value after reset = 00000000h | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DN | | | | | FN | | | RG | | | | | | Rsv | |
| Default value after reset = 00000000h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 31** | PCI | **PCI configuration register access enable.** When set to a '1', host CPU I/O cycles in address range CFCh-CFFh are converted to configuration cycles on the PCI bus. Otherwise if set to a '0', I/O cycles in this address range pass through as normal I/O cycles on the PCI bus. |
| **Bits 30-24** | Rsv | **Reserved.** Must be written to '0'. Read back as '0'. |
| **Bits 23-16** | BN | **Bus Number.** This field selects a specific bus number in the system. Bus Number 0 is assigned to the PCI bus directly behind the North Bridge. This field is driven on bits 23-16 of the AD bus during the address phase. |
| **Bits 15-11** | DN | **Device Number.** This field selects a specific device on the bus. During a Type-0 configuration cycle, this field is decoded to assert the appropriate ID-SEL line as follows;<br>The North Bridge Device Number 0xBh, which corresponds to IDSEL on AD11.<br>The South Bridge Device Number 0xCh, which corresponds to IDSEL on AD12. |
| **Bits 10-8** | FN | **Function Number.** During a PCI configuration cycle, this field is driven on bits 10-8 of the AD bus of the PCI during the address phase.<br>Function 0: PCI to ISA bridge,<br>Function 1, IDE controller. |
| **Bits 7-2** | RG | **Register Number.** During a PCI configuration cycle, this field is driven on bits 7-2 of the AD bus during the address phase. |
| **Bits 1-0** | Rsv | **Reserved**. Must be written to a '0'. Reads back as '0'. |

### 9.4. CONFIGURATION DATA REGISTER

This is a 32-bit register accessible via 8-bit, 16-bit and 32-bit IO read and write cycles.

*Config_Data* Access = 0xCFCh Regoffset =

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |

### 9.5. NORTH BRIDGE CONFIGURATION REGISTERS

The STPC North Bridge configuration registers are accessed using the values below :

Bus = 0h
Device = 0Bh (IDSEL internally connected to PCI address line 11)
Function = 0h (Host Bridge / PCI)

For example: Writing 80005800h at CF8h will access Vendor ID reg. index.

**Table 9-3. North Bridge Reset Values**

| 31 ---------- 16 | | 15 ---------- 0 | | |
|---|---|---|---|---|
| Device ID : 020Ah | | Vendor ID : 104Ah | | 00h |
| Status : 0280h | | Command : 0007h | | 04h |
| Base class code: 00h | Sub class code: 00h | Program. Inter. Reg. : 00h | Revision : 00h | 08h |
| | Header Type: 00h | | | 0Ch |
| | | | | ... |
| Control Register : 00000000h | | | | 50h |
| Error Status Register : 00000000h | | | | 54h |

### *9.5.1. NORTH BRIDGE PCI COMMAND REGISTER*

This is the 16-bit PCI command register.

*NB_Com* Access = 0xCF8h/0xCFCh Regoffset = 0x4h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | S | AD | P | VGA | MW | ES | BS | ME | IO E |
| Default value after reset = 0007h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 15-9** | **Rsv** | **Reserved**. These bits are hardwired to ‘0’. Writes have no effect on them. |
| **Bit 8** | **S** | **SERR# enable.** If this bit is set to a ‘1’, the North Bridge may assert SERR# upon detecting a target abort in response to an North Bridge initiated PCI transaction, upon being forced to end a non-configuration space transaction with a master abort, or if a parity error on the PCI bus is detected. If this bit is set to ‘0’, the North Bridge will not assert SERR#. |
| **Bit 7** | **AD** | **Address/Data stepping enable.** This bit is hardwired to a ‘0’. Writes to it have no effect. |
| **Bit 6** | **P** | **PERR# response.** Must always be set to ’0’. |
| **Bit 5** | **VGA** | **VGA Palette Snoop enable.** This bit is hardwired to a ‘0’. Writes to it have no effect. |
| **Bit 4** | **MW** | **Master Write and Invalidate Enable.** This bit is hardwired to a ‘0’. Writes to it have have no effect. |
| **Bit 3** | **ES** | **Enable Special cycles.** This bit is hardwired to a ‘0’. Writes to it have no effect. |
| **Bit 2** | **BS** | **Bus Master enabled.** This bit is hardwired to a ‘1’. Writes to it have no effect. |
| **Bit 1** | **ME** | **Memory Enable.** This bit is hardwired to a ‘1’. Writes to it have no effect. |
| **Bit 0** | **I/O E** | **I/O Enable.** This bit is hardwired to a ‘1’. Writes to it have no effect. |

### *9.5.2. NORTH BRIDGE PCI STATUS REGISTER*

This is the 16-bit PCI Status register.

*NB_Stat* Access = 0xCF8h/0xCFCh Regoffset = 0x6h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DP | SS | SMA | RTA | STA | DT | | DPED | FBBC | Rsv | | | | | | |
| Default value after reset = 0280h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 15** | **DP** | **Detected parity error.** This bit is set when a PCI parity error is detected. It may be cleared by software by writing a '1' back to this bit. |
| **Bit 14** | **SS** | **Signalled SERR#.** This bit is set to a '1' when SERR# is asserted by the North Bridge. Writing a '1' to this bit will clear it. |
| **Bit 13** | **SMA** | **Signalled Master Abort.** This bit is set to a 1 when the North Bridge terminates a PCI transaction with a master abort. Writing a '1' to this bit will clear it. |
| **Bit 12** | **RTA** | **Received Target Abort.** This bit is set to a '1' when PCI transaction initiated by the North Bridge is terminated with a target abort. Writing a '1' to this bit will clear it. |
| **Bit 11** | **STA** | **Signalled Target Abort.** This bit is hardwired to '0'. |
| **Bits 10-9** | **DT** | **DEVSEL Timing.** These bits are hardwired for medium timing to '01'. Writes have no effect. |
| **Bit 8** | **DPED** | **Data Parity Error Detected.** This bit is set to '1' when a PCI data parity error is detected. Writing a '1' will clear it. |
| **Bit 7** | **FBBC** | **Fast Back-to-Back Capable.** Hardwired to '1'. Indicates that the North Bridge, while acting as target, is capable of accepting fast back-to-back transactions. Reads will always return '1', writes have no effect. |
| **Bits 6-0** | **Rsv** | **Reserved.** These bits are hardwired to '0's. |

### *9.5.3. NORTH BRIDGE PCI REVISION ID REGISTER*

This is the 8-bit read only PCI revision identification register.

*NB_R_ID* Access = 0xCF8h/0xCFCh Regoffset = 0x8h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-0** | **Rsv** | **Reserved.** These bits are hardwired to 00h. |

### *9.5.4. NORTH BRIDGE DEVICE CLASS CODE REGISTER*

This is a 24-bit read only register implemented at configuration space offset 9h, Ah, Bh.

*NB_C_Code* Access = 0xCF8h/0xCFCh Regoffset = 0x9h

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BCC | | | | | | | | SCC | | | | | | | |
| Default value after reset = 00h | | | | | | | | Default value after reset = 00h | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| PIR | | | | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 31-24** | **BCC** | **Base Class Code.** These bits are hardwired to 00h. |
| **Bits 23-16** | **SCC** | **Sub Class Code.** These bits are hardwired to 00h. |
| **Bits 15-8** | **PIR** | **Programming Interface Register.** These bits are hardwired to 00h. |

*9.5.5. NORTH BRIDGE HEADER TYPE REGISTER*

This is an 8-bit read only register, hardwired to 00h

.

*NB_Head* Access = 0xCF8h/0xCFCh Regoffset = 0xEh

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *9.5.6. NORTH BRIDGE CONTROL REGISTER*

*NB_Cont* Access = 0xCF8h/0xCFCh Regoffset = 0x50h

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | | | PCI1 | PCI2 | PCI3 | Rsv | | | |
| Default value after reset = 00000000h | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | | | | | P | | | SP | S |
| Default value after reset = 00000000h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 31-23** | **Rsv** | **Reserved.** Hardwired to ‘0’. |
| **Bit 22** | **PCI1** | **PCI 2.0 Enable.** If this bit is set to ‘1’, North Bridge will be compatible with PCI 2.0 standard. If this bit is set to ‘0’, North Bridge is compatible with PCI 2.1 standard. |
| **Bit 21** | **PCI2** | **PCI to Host Read Prefetch Enable.** If this bit is set to ‘1’, all QWORD aligned burst reads from a PCI master addressed to the North Bridge system memory will use prefetch. If set to ‘0’, memory read cycles from PCI to host are allowed to complete before the PCI cycle is terminated and all burst read attempts will be disconnected on the PCI bus. |
| **Bit 20** | **PCI3** | **PCI to Host Write Posting Enable.** If this bit is set to ‘1’, all burst writes from a PCI master addressed to the North Bridge system memory will be posted. If it is set to ‘0’, all memory write cycles from PCI to host are allowed to complete before the PCI cycle is terminated and all burst write attempts will be disconnected on the PCI bus. |
| **Bits 19-5** | **Rsv** | **Reserved.** Hardwired to ‘0’. |
| **Bit 4** | **P** | **PERR_ on read data parity error enable.** |
| **Bit 3** | **P** | **PERR_ on write data parity error enable.** |
| **Bit 2** | **P** | **PERR_ on address parity error enable.** |
| **Bit 1** | **SP** | **SERR_ on PERR_ enable.** |
| **Bit 0** | **S** | **SERR_ on received target abort.** |

### *9.5.7. NORTH BRIDGE PCI ERROR STATUS REGISTER*

*NB_E_Stat* Access = 0xCF8h/0xCFCh Regoffset = 0x54h

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | | | | | | | | | |
| Default value after reset = 00000000h | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | | | | | RDP | WDP | AP | PES | RTAE |
| Default value after reset = 00000000h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 31-5** | **Rsv** | **Reserved.** Hardwired to '0'. |
| **Bit 4** | **RDP** | **Read Data Parity Error Status.** This bit is set when a PCI read data parity error is detected. Writing a '1' will clear it. |
| **Bit 3** | **WDP** | **Write Data Parity Error Status.** This bit is set when a PCI write data parity error is detected. Writing a '1' will clear it. |
| **Bit 2** | **AP** | **Address Parity Error Status.** This bit is set when a PCI address parity error is detected. Writing a '1' will clear it. |
| **Bit 1** | **PES** | **Parity Error Status.** System errors as a result of a parity error status. This bit is set to '1' when SERR# was asserted as a result of parity error. Writing a '1' will clear it. |
| **Bit 0** | **RTAE** | **Received Target Abort Error.** System errors as a result of a received target abort. This bit is set to '1' when SERR# was asserted as a result of receiving a target abort. Writing a '1' will clear it. |

### 9.6. THE SOUTH BRIDGE

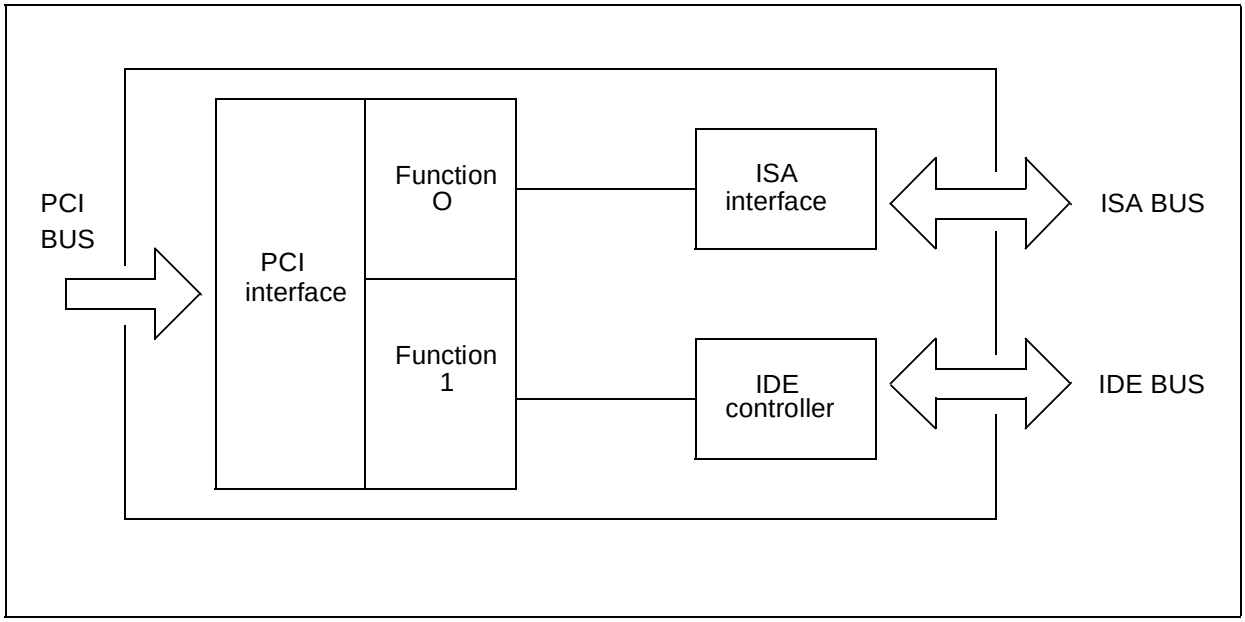



**Figure 9-2. South Bridge Layout**

Figure 9-2 Illustrates the South Bridge layout and the associated functions.

The STPC South Bridge configuration registers are accessed using the values below :

Bus = 0
Device = Ch (IDSEL internally connected to PCI address line 12)
Function = 0 (ISA bridge)
- Responds to I/O / memory / config
- Translates Master ISA to PCI
- Translates PCI to Slave ISA

Function = 1 (IDE controller)
- Responds to IO / config

For example: Writing 80006110h at CF8h will access Function 1 (IDE) Command reg. index.

## 9.7. SOUTH BRIDGE PCI FUNCTION 0 (PCI to ISA) CONFIGURATION REGISTERS

**Table 9-4. Function 0 (ISA Bridge) Configuration Space Register Reset Values**

| 31 | | 16 15 | 0 | |
|---|---|---|---|---|
| Device ID: 021Ah | | Vendor ID: 104Ah | | 00h |
| Status: 0280h | | Command: 000Fh | | 04h |
| Base class code: 06h | Sub class code: 01h | Program. Inter. Reg. : 00h | Revision ID: 00h | 08h |
| | Header: 80h | | | 0Ch |
| | | | | ... |
| | | | | ... |
| | | | Miscellaneous reg : 00h | 40h |

This section describes Function 0 (F#0) configuration registers, including the PCI to ISA bridge control. The registers and reset values are illustrated in Table 9-4.

### *9.7.1. SOUTH BRIDGE PCI COMMAND REGISTER*

This is the 16-bit PCI command register.

*SB_Com_0* Access = 0xCF8h/0xCFCh Regoffset = 0x4h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | S | AD | P | VGA | MW | ESC | BM | ME | IO E |
| Default value after reset = 000Fh | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 15-9** | **Rsv** | **Reserved**. These bits are hardwired to '0's. Writes have no effect on them. |
| **Bit 8** | **S** | **SERR# enable.** If this bit is set to a '1', the South Bridge may assert SERR# upon detecting a target abort in response to a South Bridge initiated PCI transaction on behalf of an ISA master. If this bit is set to '0', the South Bridge will not assert SERR#. |
| **Bit 7** | **AD** | **Address/Data stepping enable.** This bit is hardwired to a '0'. Writes to it have no effect. |
| **Bit 6** | **P** | **PERR# response.** Setting this bit to '1' enables parity error detection. |
| **Bit 5** | **VGA** | **VGA Palette Snoop enable.** This bits is hardwired to a '0'. Writes to it have no effect. |
| **Bit 4** | **MW** | **Master Write and Invalidate Enable.** This bit is hardwired to a '0'. Writes to it have no effect. |
| **Bit 3** | **ESC** | **Enable Special Cycles.** This bit is hardwired to a '1'. The South Bridge writes to it have no effect. The South Bridge responds to halt and shutdown cycles. |
| **Bit 2** | **BM** | **Bus Master enabled.** This bit is hardwired to a '1'. Writes to it have no effect. |
| **Bit 1** | **ME** | **Mem Enable.** This bit is hardwired to a '1'. Writes to it have no effect. |
| **Bit 0** | **I/O E** | **I/O Enable.** This bit is hardwired to a '1'. Writes to it have no effect. |

### *9.7.2. SOUTH BRIDGE PCI STATUS REGISTER*

This is the 16-bit PCI Status register.

*SB_Stat0* Access = 0xCF8h/0xCFCh Regoffset = 0x6h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsv | SS | SMA | RTA | STA | DT | | DPED | FBBC | Rsv | | | | | | |
| Default value after reset = 0280h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 15** | **Rsv** | **Reserved**. This bit is hardwired to '0'. |
| **Bit 14** | **SS** | **Signalled SERR#.** This bit is set to a '1' when SERR# is asserted by the South Bridge on behalf of an ISA master cycle. Writing a '1' to this bit will clear it. SERR# is asserted in response to a target abort during an ISA master cycle on PCI bus and if bit-8 of the F#0 PCI command register is set to a '1' to enable SERR# signalling. |
| **Bit 13** | **SMA** | **Signalled Master Abort.** This bit is hardwired to a '0'. |
| **Bit 12** | **RTA** | **Received Target Abort.** This bit is set to a '1' when the PCI transaction is initiated by the South Bridge on behalf of an ISA master is terminated with a target abort. Writing a '1' to this bit will clear it. |
| **Bit 11** | **STA** | **Signalled Target Abort.** This bit is set to a '1' when the South Bridge terminates a PCI transaction with a target abort. Writing a '1' to this bit will clear it. The South Bridge will generate target abort if a A1-0 of a PCI IO cycle does not match the Byte enables. |
| **Bits 10-9** | **DT** | **DEVSEL Timing.** These bits are hardwired for medium timing to '01'. Writes have no effect. |
| **Bit 8** | **DPED** | **Data Parity Error Detected.** This bit is hardwired to '0'. |
| **Bit 7** | **FBBC** | **Fast Back-to-Back Capable.** Hardwired to '1'. Indicates that the South Bridge, while acting as target, is capable of accepting fast back-to-back transactions. Reads will always return '1', writes have no effect. |
| **Bits 6-0** | **Rsv** | **Reserved.** These bits are hardwired to '0'. |

### *9.7.3. SOUTH BRIDGE PCI REVISION ID REGISTER*

This is the 8-bit read only PCI revision identification register.

*SB_R_ID0* Access = 0xCF8h/0xCFCh Regoffset = 0x8h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 7-0: These bits are hardwired to 00h.

### *9.7.4. SOUTH BRIDGE DEVICE CLASS CODE REGISTER*

This is a 24-bit read only register implemented at configuration space offset 09h, 0Ah, 0Bh.

*SB_C_Code0* Access = 0xCF8h/0xCFCh Regoffset = 0x9h

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BCC | | | | | | | | SCC | | | | | | | |
| Default value after reset = 06h | | | | | | | | Default value after reset = 01h | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| PIR | | | | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 31-24** | **BCC** | **Base Class Code.** These bits are hardwired to 06h (Bridge Device). |
| **Bits 23-16** | **SCC** | **Sub Class Code.** These bits are hardwired to 01h (ISA Bridge). |
| **Bits 15-8** | **PIR** | **Programming Interface Register.** These bits are hardwired to 00h. |

### *9.7.5. SOUTH BRIDGE HEADER TYPE REGISTER*

This register is hardwired to 80h, indicating that the South Bridge is a multi-function PCI device.

*SB_Head0* Access = 0xCF8h/0xCFCh Regoffset = 0xEh

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *9.7.6. SOUTH BRIDGE MISCELLANEOUS REGISTER*

*SB_Misc0* Access = 0xCF8h/0xCFCh Regoffset = 040h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | | | GPIO_MDC | GPIO_SDC | GPIO | PCI |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-4** | **Rsv** | **Reserved.** Hardwired to 00h. |
| **Bit 3** | **GPIO_MDC** | **GPIO Master Debounce Control :** enable/disable the debounce logic on master GPIO<br>0 = Disabled<br>1 = Enabled |
| **Bit 2** | **GPIO_SDC** | **GPIO Slave Debounce Control**: enable/disable the debounce logic on slave GPIO<br>0 = Disabled<br>1 = Enabled |
| **Bits 1** | **GPIO** | GPIO[1)] Enable/Disable: enable/disable all GPIOs<br>0 = Disabled<br>1 = Enabled |
| **Bit 0** | **PCI** | **PCI 2.0 Enable.** If this bit is set to '1', South Bridge will be compatible with PCI 2.0 standard. If this bit is set to '0', South Bridge is compatible with PCI 2.1 standard. |

Note 1. Enabling GPIO will enable both master and slave GPIO controllers.

Programming Notes; This register is used to initiallise the GPIOs as well as select the mode in which the Bridge operates. For information on how to initialise the GPIOs, refer to Table 13-1

## 9.8. SOUTH BRIDGE PCI FUNCTION 1 CONFIGURATION REGISTERS

This section describes the Function 1 (F#1) configuration registers. The registers and reset values are illustrated in Table 7-16.

**Table 9-5. Function 1 (IDE Bridge) PCI Configuration Space Register Reset Values**

| 31 | 16 | 15 | 0 | |
|---|---|---|---|---|
| Device: | | Vendor ID: 104Ah | | 00h |
| Status: 0280h | | Command: 0000h | | 04h |
| Base class code: 01h | Sub class code: 01h | Program. Inter. Reg. : 8Ah | Revision ID: 00h | 08h |
| | Header: 80h | Reserved: 00h | | 0Ch |
| IO Base address 0 register: 00000001h | | | | 10h |
| IO Base address 1 register: 00000001h | | | | 14h |
| IO Base address 2 register: 00000001h | | | | 18h |
| IO Base address 3 register: 00000001h | | | | 1Ch |
| Reserved | | | | 20h |
| | | | | ... |
| | | | | ... |
| Primary IDE Timing register: 97609760h | | | | 40h |
| Secondary IDE Timing register: 97609760h | | | | 44h |
| | | | Miscellaneous reg : 00h | 48h |

### *9.8.1. SOUTH BRIDGE VENDOR IDENTIFICATION REGISTER*

This is a 16-bit read-only register implemented at configuration space offset 00h and 01h. It contains the Vendor Identifier assigned to STPC.

Bits 15-0: These bits are hardwired to 100Eh.

*SB_V_ID1* Access = 0xCF8h/0xCFCh Regoffset = 0x0h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

Writes to this register have no effect.

### *9.8.2. SOUTH BRIDGE DEVICE IDENTIFICATION REGISTER*

This is a 16-bit read only register implemented at configuration space offset 02h and 03h. It contains the Device Identifier assigned to the South Bridge.

Bits 15-0: These bits are hardwired to 55CCh

*SB_D_ID1* Access = 0xCF8h/0xCFCh Regoffset = 0x2h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |

Writes to this register have no effect.

### *9.8.3. SOUTH BRIDGE PCI COMMAND REGISTER*

This is the 16-bit PCI command register.

*SB_Com1* Access = 0xCF8h/0xCFCh Regoffset = 0x4h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | SE | A | P | VGA | MWIE | ESC | Rsv | M E | IO E |
| Default value after reset = 0000h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 15-9** | **Rsv** | **Reserved.** These bits are hardwired to ‘0’s. Writes have no effect on them. |
| **Bit 8** | **SE** | **SERR# Enable.** If this bit is set to a ‘1’, the South Bridge may assert SERR# upon detecting a master or target abort in response to a the South Bridge initiated PCI transaction on behalf of IDE master. If this bit is set to ‘0’, the South Bridge will not assert SERR#. |
| **Bit 7** | **A** | **Address/Data stepping enable.** This bit is hardwired to a ‘0’. Writes to it have no effect. |
| **Bit 6** | **P** | **PERR# response.** Setting this bit to ‘1’ enables parity error detection. |
| **Bit 5** | **VGA** | **VGA Palette Snoop enable.** This bits is hardwired to a ‘0’. Writes to it have no effect. |
| **Bit 4** | **MWIE** | **Master Write and Invalidate Enable.** This bit is hardwired to a ‘0’. Writes to it have have no effect. |
| **Bit 3** | **ESC** | **Enable Special cycles.** This bit is hardwired to a ‘0’. Writes to it have no effect. |
| **Bit 2** | **Rsv** | **Reserved.** |
| **Bit 1** | **M E** | **Mem Enable.** This bit is hardwired to a ‘0’. Writes to it have no effect. |
| **Bit 0** | **IO E** | **IO Enable.** Setting this bit to a ‘1’ enables access to the IDE IO registers. |

### *9.8.4. SOUTH BRIDGE PCI STATUS REGISTER*

*SB_Stat1* Access = 0xCF8h/0xCFCh Regoffset = 0x6h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsv | SS | SMA | RTA | Rsv | DT | | DPED | FBBC | Rsv | | | | | | |
| Default value after reset = 0280h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 15** | **Rsv** | **Reserved**. This bit is hardwired to ‘0’. |
| **Bit 14** | **SS** | **Signalled SERR#.** This bit is set to a ‘1’. |
| **Bit 13** | **SMA** | **Signalled Master Abort.** This bit is set to a ‘1’ when the West Bridge terminates a PCI transaction initiated on behalf of the IDE master with a master abort. The West Bridge master aborts an IDE master cycle if no target responds to this cycle. |
| **Bit 12** | **RTA** | **Received Target Abort.** This bit is set to a ‘1’ when a PCI transaction initiated by the West Bridge on behalf of the IDE master is terminated with a target abort. Writing a ‘1’ to this bit will clear it. |
| **Bit 11** | **Rsv** | **Reserved**. This bit is hardwired to ‘0’. |
| **Bits 10-9** | **DT** | **DEVSEL Timing.** These bits are hardwired for medium timing to ‘01’. Writes to these bits have no effect. |
| **Bit 8** | **DPED** | **Data Parity Error Detected.** This bit is hardwired to ‘0’. |
| **Bit 7** | **FBBC** | **Fast Back-to-Back Capable.** This bit is hardwired to ‘1’. |
| **Bits 6-0** | **Rsv** | **Reserved**. These bits are hardwired to ‘0’. |

### *9.8.5. SOUTH BRIDGE REVISION ID REGISTER*

This is the 8-bit read only PCI revision identification register.

*SB_R_ID1* Access = 0xCF8h/0xCFCh Regoffset = 0x8h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-0** | **Rsv** | **Reserved.** These bits are hardwired to 00h in this stepping of the chip. |

### *9.8.6. SOUTH BRIDGE PROGRAMMING INTERFACE REGISTER*

*Prog_Int* Access = 0xCF8h/0xCFCh Regoffset = 0x09h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | Rsv | | | | | | |
| Default value after reset = 8Ah | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **Rsv** | **Reserved.** This bit is hardwired to '1'. Writes to have no effect on this bit. |
| **Bits 6-4** | **Rsv** | **Reserved.** These bits are hardwired to '0'. These bits are hardwired to '0'. |
| **Bit 3** | | This bit is hardwired to '1' indicating that the secondary channel is programmable to be either in legacy or native mode. |
| **Bit 2** | | This bit selects the operating mode of the secondary channel. (see Table 7-17) |
| **Bit 1** | | This bit is hardwired to '1' indicating that the primary channel is programmable to be either in legacy or native mode. |
| **Bit 0** | | This bit selects the operating mode of the primary channel. (see Table 7-18) |

**Table 9-6. Operating Mode of the Secondary Channel**

| Bit 2 | |
|---|---|
| 0 | Channel is in legacy mode. In legacy mode the secondary IDE channel occupies IO addresses 170h-177h and 376h. |
| 1 | Channel is in native mode. The address range occupied by the secondary IDE controller in native mode is specified by base address registers 2 and 3. |

**Table 9-7. Operating Mode of the Primary Channel**

| Bit 0 | |
|---|---|
| 0 | Channel is in legacy mode. In legacy mode the Primary IDE channel occupies IO addresses 1F0h-1F7h and 3F6h. |
| 1 | Channel is in native mode. The address range occupied by the Primary IDE controller in native mode is specified by base address registers '0' and '1'. |

### *9.8.7. SOUTH BRIDGE SUB-CLASS CODE REGISTER*

This register is hardwired to 01h, indicating that this is an IDE controller device.

*Sub_Class* Access = 0xCF8h/0xCFCh Regoffset = 0xAh

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### *9.8.8. SOUTH BRIDGE BASE-CLASS CODE REGISTER*

This register is hardwired to 01h, indicating that Function 1 is a mass storage device.

*Base_Class* Access = 0xCF8h/0xCFCh Regoffset = 0xBh

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### *9.8.9. SOUTH BRIDGE LATENCY TIMER CONTROL REGISTER*

*Lat_T* Access = 0xCF8h/0xCFCh Regoffset = 0xDh

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-0** | **Rsv** | **Reserved.** These bits are hardwired to ‘0’. |

### *9.8.10. SOUTH BRIDGE HEADER TYPE REGISTER*

This register is hardwired to 80h, indicating that the South Bridge is a PCI multi-function device.

*Head_T* Access = 0xCF8h/0xCFCh Regoffset = 0xEh

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### *9.8.11. SOUTH BRIDGE IDE BASE ADDRESS 0 REGISTER*

This 32-bit register contains the base IO address for accessing the primary IDE channel command registers. The base address is meaningful only when the Primary channel is programmed for native mode operation. If programmed for legacy mode operation, the primary channel command registers are decoded at 1F0h IO address.

*Base0* Access = 0xCF8h/0xCFCh Regoffset = 0x10h

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BA | | | | | | | | | | | | | | | |
| Default value after reset = 00000001h | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BA | | | | | | | | | | | | | | Rsv | MSI |
| Default value after reset = 00000001h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 31-3** | **BA** | **Base Address.** This field specifies the 8-Byte IO address range where the primary channel command registers are located. |
| **Bit 2** | | Hardwired to ‘0’ to indicate that this base address occupies 4-Bytes in IO space. |
| **Bit 1** | **Rsv** | **Reserved**. Hardwired to ‘0’. |
| **Bit 0** | **MSI** | **Memory Space Indicator.** This bit is hardwired to ‘1’ to indicate IO space. |

### *9.8.12. SOUTH BRIDGE IDE BASE ADDRESS 1 REGISTER*

This 32-bit register contains the base IO address for accessing the primary IDE channel Control registers. The base address is meaningful only when the Primary channel is programmed for native mode operation. If programmed for legacy mode operation, the primary channel control register are decoded at 3F6h.

*Base1* Access = 0xCF8h/0xCFCh Regoffset = 0x14h

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BA | | | | | | | | | | | | | | | |
| Default value after reset = 00000001h | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BA | | | | | | | | | | | | | | Rsv | MSI |
| Default value after reset = 00000001h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 31-2** | **BA** | **Base Address.** This field specifies the 4-Byte IO address range where the primary channel command registers are located. |
| **Bit 1** | **Rsv** | **Reserved**. Hardwired to ‘0’. |
| **Bit 0** | **MSI** | **Memory Space Indicator.** This bit is hardwired to ‘1’ to indicate IO space. |

### *9.8.13. SOUTH BRIDGE IDE BASE ADDRESS 2 REGISTER*

This 32-bit register contains the base IO address for accessing the secondary IDE channel command registers. The base address is meaningful only when the secondary channel is programmed for native mode operation. If programmed for legacy mode operation, the secondary channel command registers are decoded at 170h IO address.

*Base2* Access = 0xCF8h/0xCFCh Regoffset = 0x18h

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BA | | | | | | | | | | | | | | | |
| Default value after reset = 00000001h | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BA | | | | | | | | | | | | | | Rsv | MSI |
| Default value after reset = 00000001h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 31-3** | **BA** | **Base Address.** This field specifies the 8-Byte IO address range where the secondary channel command registers are located. |
| **Bit 2** | | Hardwired to ‘0’ to indicate that this base address occupies 4-Bytes in IO space. |
| **Bit 1** | **Rsv** | **Reserved**. Hardwired to ‘0’. |
| **Bit 0** | **MSI** | **Memory Space Indicator.** This bit is hardwired to ‘1’ to indicate IO space. |

### *9.8.14. SOUTH BRIDGE IDE BASE ADDRESS 3 REGISTER*

This 32-bit register contains the base IO address for accessing the secondary IDE channel Control registers. The base address is meaningful only when the secondary channel is programmed for native mode operation. If programmed for legacy mode operation, the secondary channel control register is decoded at 376h.

*Base3* Access = 0xCF8h/0xCFCh Regoffset = 0x1Ch

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BA | | | | | | | | | | | | | | | |
| Default value after reset = 00000001h | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BA | | | | | | | | | | | | | | Rsv | MSI |
| Default value after reset = 00000001h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 31-2** | **BA** | **Base Address.** This field specifies the 4-Byte IO address range where the secondary channel command registers are located. |
| **Bit 1** | **Rsv** | **Reserved**. Hardwired to ‘0’. |
| **Bit 0** | **MSI** | **Memory Space Indicator.** This bit is hardwired to ‘1’ to indicate IO space. |

*9.8.15. SOUTH BRIDGE IDE BASE ADDRESS 4 REGISTER*

This 32-bit register contains the base IO address for accessing the bus master control and status register.

*Base4* Access = 0xCF8h/0xCFCh Regoffset = 0x20h

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BA | | | | | | | | | | | | | | | |
| Default value after reset = 00000001h | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | HW | | Rsv | MSI |
| Default value after reset = 00000001h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 31-4** | **BA** | **Base Address.** This field specifies the 16-bytes IO address range where the Bus master control and status registers are located. |
| **Bits 3-2** | **HW** | Hardwired to ‘0’ to indicate that this base address occupies 16-bytes in IO space. |
| **Bit 1** | **Rsv** | **Reserved**. Hardwired to ’0’. |
| **Bit 0** | **MSI** | **Memory space indicator.** This bit is hardwired to ‘1’ to indicate IO space. |

### *9.8.16. SOUTH BRIDGE IDE TIMING REGISTER*

This 16-bit register contains all the IDE timing information for the Read and Write signals. This register is duplicated with the appropriate offsets for the Primary Master, Slave, Secondary Master or Slave Timings. The offsets are shown in Table 7-19

**Table 9-8. Timing Register Location**

| Regoffset | Timing Register |
|---|---|
| 040h | Primary Master Timing Control |
| 042h | Primary Slave Timing Control |
| 044h | Secondary Master Timing Control |
| 046h | Secondary Slave Timing Control |

*IDE_Timing* Access = 0xCF8h/0xCFCh Regoffset = Table 7-19

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IDSMS | | IDRT | | IDAT | | IPRT | | | IPAT | | | ISE | EWP | ERP | EPA |
| Default value after reset = 7F60h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 15-14** | **IDSMS** | **IDE DMA Speed Mode Select.** These bits, along with bits 29-26, determine the width of the read and write signals during DMA transfers. Refer to Table 7-20 to determine the number of clocks for active and recovery times for the read/write signals.<br>The slower modes are normally used for single word DMA modes and the faster modes are used for double word DMA modes. |
| **Bits13-12** | **IDRT** | **IDE DMA Recovery Time.** These bits, along with bits 31-30, determine the duration of the recovery (inactive) time of the read/write signals during DMA transfers. Refer to Table 7-21 to determine the number of clocks for recovery times of the read/write signals. Default is 01h. |
| **Bits 11-10** | **IDAT** | **IDE DMA Active Time.** These bits, along with bits 31-30, determine the duration of the active time of the read/write signals during DMA transfers. Refer to Table 7-22 to determine the number of clocks for active times of the read/write signals. Default is 01h. |
| **Bits 9-7** | **IPRT** | **IDE PIO Recovery Time.** These bits determine the duration of the recovery (inactive) time of the read/write signals during PIO data transfers.<br>The transfers to non-data registers of the IDE device will always have a recovery time of 10 clocks.<br>Refer to Table 7-23 to determine the number of clocks for recovery times of the read/write signals. |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 6-4** | **IPAT** | **IDE PIO Active Time.** These bits determine the duration of the active time of the read/write signals during PIO transfers.<br>The transfers to non-data registers of the IDE device will always have an active time of 10 clocks.<br>Refer to Table 7-24 to determine the number of clocks for active times of the read/write signals:<br>Note that the address setup time is implied. After initial start-up latency, the address setup time is shown in Table 7-25 |
| **Bit 3** | **ISE** | **IOCHRDY Sampling Enable.** This bit when set to 1, enables IOCHRDY sampling of the IDE device, i.e., the active read/write signals will be stretched (when IOCHRDY is low), to extend the cycle, if this bit is set. When set to 0, IOCHRDY is ignored and the read/write signals are deasserted after the specified number of PCI clocks. |
| **Bit 2** | **EWP** | **Enable Write Posting.** This bit when set to 1, enables posting of data into the FIFO during PIO data write transfers. Note that the non-data writes are never posted. |
| **Bit 1** | **ERP** | **Enable Read Prefetch.** This bit when set to 1 enables data to be prefetched into the data FIFO during PIO read commands. If set to 0, prefetch is completely disabled. |
| **Bit 0** | **EPA** | **Enable Prefetch for ATAPI commands.** When set to 1, this bit enables prefetching for ATAPI commands (when A0h is written into the command register - Register offset 07h for the device). When set to 0, prefetching during ATAPI commands is disabled. This is to accommodate non-512 boundary data transfers that are supported by ATAPI devices.<br>Prefetch for the IDE controller is given in Table 7-26. |

**Table 9-9. DMA Speed Mode Select**

| Bit 15 | Bit 14 | DMA Speed Mode |
|---|---|---|
| 0 | 0 | Fast mode |
| 0 | 1 | Medium fast mode |
| 1 | 0 | Medium slow |
| 1 | 1 | Slow mode |

**Table 9-10. IDE DMA Recovery Time Settings**

| | | Bits 13-12 | Bits 13-12 | Bits 13-12 | Bits 13-12 |
|---|---|---|---|---|---|
| **Bit 15** | **Bit 14** | 00 | 01 | 10 | 11 |
| 0 | 0 | 1 | 2 | 3 | 4 |
| 0 | 1 | 5 | 6 | 7 | 8 |
| 1 | 0 | 9 | 10 | 11 | 12 |
| 1 | 1 | 14 | 15 | 16 | 20 |

**Table 9-11. IDE DMA Active Time Settings**

| | Bits 11-10 | Bits 11-10 | Bits 11-10 | Bits 11-10 |
|---|---|---|---|---|
| **Bits 15-14** | 00 | 01 | 10 | 11 |
| 00 | 1 | 2 | 3 | 4 |
| 01 | 5 | 6 | 7 | 8 |
| 10 | 9 | 10 | 11 | 12 |
| 11 | 14 | 15 | 16 | 20 |

**Table 9-12. Recovery R/W Signal Time**

| Bit 9 | Bit 8 | Bit 7 | PCI Clocks |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 3 |
| 0 | 1 | 1 | 4 |
| 1 | 0 | 0 | 8 |
| 1 | 0 | 1 | 9 |
| 1 | 1 | 0 | 10 |
| 1 | 1 | 1 | 12 |

**Table 9-13. Active R/W Signal Time**

| Bit 6 | Bit 5 | Bit 4 | PCI Clocks |
|---|---|---|---|
| 0 | 0 | 0 | 2 |
| 0 | 0 | 1 | 3 |
| 0 | 1 | 0 | 4 |
| 0 | 1 | 1 | 5 |
| 1 | 0 | 0 | 8 |
| 1 | 0 | 1 | 9 |
| 1 | 1 | 0 | 10 |
| 1 | 1 | 1 | 12 |

**Table 9-14. Address Setup Time**

| Bit 6 | Bit 5 | Address setup time |
|---|---|---|
| 0 | 0 | 1 clock |
| 0 | 1 | 2 clocks |
| 1 | X | 3 clocks |

**Table 9-15. Prefetch Encoding**

| Bit 1 | Bit 0 | Prefetch |
|---|---|---|
| 0 | X | Completely disabled |

**Table 9-15. Prefetch Encoding**

| Bit 1 | Bit 0 | Prefetch |
|---|---|---|
| 1 | 0 | Enabled for non-ATAPI commands only |
| 1 | 1 | Enabled for all commands |

### *9.8.17. SOUTH BRIDGE MISCELLANEOUS REGISTER*

This register contains miscellaneous informations.

*SB_Misc1* Access = 0xCF8h/0xCFCh Regoffset = 0x48h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SR | Rsv | Rsv | Rsv | Rsv | Rsv | SID | PID |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **SR** | **Soft Reset.** When set to 1, the IDE controller is reset. It does not affect the timing control register. The FIFOs and the internal state machines are cleared. |
| **Bits 6-2** | **Rsv** | **Reserved.** |
| **Bit 1** | **SID** | **Secondary Interrupt Detect.** This bit is set when the secondary interrupt is active. It is cleared by writing a 1 to this bit in the register. |
| **Bit 0** | **PID** | **Primary Interrupt Detect.** This bit is set when the primary interrupt is active. It is cleared by writing a 1 to this bit in the register. |

# 10. ISA INTERFACE

## 10.1. INTRODUCTION

The ISA Interface provides access to the peripherals available in the STPC device and to Memory and external devices on the ISA bus.

Control of the ISA bus is by the North Bridge which acts as a bridge between the host CPU bus and the PCI bus. Reads and writes which are initiated by the CPU are subtractively decoded. Reads and writes that target North Bridge internal registers or main memory are routed to those targets, and all other reads and writes are sent to the PCI bus. The cycles for interrupt acknowledge, shutdown, stop grant and halt are also sent to the PCI bus.

The North Bridge also routes PCI reads and writes to cache, main memory and its internal registers. The South Bridge acts as a bridge between the PCI bus and the ISA bus. ISA bus cycles may be initiated by PCI bus cycles, or by an ISA bus card. Additionally, refresh cycles are run periodically by the ISA controller.

The South Bridge will claim all PCI cycles which were initiated outside the South Bridge and not claimed by any other PCI slave. Reads and writes to PCI configuration registers are routed appropriately by the South Bridge's PCI controller. All other PCI operations, including reads and writes to the South Bridge internal registers, are sent to the ISA controller. With the exception of writes to the keyboard controller under certain conditions, a read or write cycle sent to the ISA bus controller will create one or more ISA bus cycles.

Because of the speed difference between ISA bus and PCI bus, and the requirement that PCI cycles be less than a certain number of clocks, PCI cycles which go to the ISA bus will require retries on the PCI bus.

The cycles for interrupt acknowledge, shutdown, stop grant and halt are also sent to the ISA bus controller. These cycles do not create ISA bus cycles, but they use the same state machines for timing and arbitration as reads and writes. ISA bus cycles which are initiated by an ISA bus card are either DMA cycles, in which case the address is supplied by the DMA controller, or ISA bus master cycles, in which case the address is supplied by the card itself.

Every cycle initiated on the ISA bus is tried on the PCI bus. If the cycle is claimed by some PCI target, then data is read from or written to that target. If the PCI cycle is not claimed, and the cycle targets a South Bridge internal register, then that register is read from or written to. Otherwise, the target is expected to be on the ISA bus.

## 10.2. PCI / ISA CYCLES

### *10.2.1. PCI TO ISA READ AND WRITE*

The PCI transfers data four bytes at a time, with byte enables for each byte. The South Bridge's PCI controller transfers these four bytes and four byte enables to the ISA controller. The ISA controller in turn runs zero to four ISA cycles. For 8-bit targets, the enabled bytes are read or written in order, least significant byte (lowest address) first.

For 16-bit targets, enabled bytes are again read or written in order, but a 16-bit transfer is used when an even byte is enabled and the following odd byte is also enabled.

Eight-bit ISA operations are by default four and a half ISACLK cycles, starting on a falling edge of ISACLK and ending on a rising edge. Sixteen-bit cycles are by default two and a half ISACLK cycles, also starting on a falling edge of ISACLK and ending on a rising clock. An additional clock cycle may be added by setting bit 5 in Index Register 50. Cycles can also be extended by pulling IOCHRDY low.

### *10.2.2. PCI TO INTERNAL REGISTER READ AND WRITE*

All South Bridge internal registers are 8-bit. If an IO read or write targets an internal register, the target is assumed to be 8-bit wide (that is IOCS16# is ignored). Timing for reads and writes to internal registers is the same as 8-bit cycles on the ISA bus (see Section 10.2.1.).

If a write targets an internal register of the South Bridge, the data is written to the register and also to the ISA bus. If a read targets an internal register, the internal register is read, the South Bridge drives the ISA data bus with the contents of the register, and a ISA read cycle is done.

Registers that are called index registers in this document are indirectly addressed through a register at IO address 22h. There are two copies of this register, one on the North Bridge and one on the South Bridge.

Writes to IO address 22h go to both copies of the register. Reads from IO address 22h normally come from the North Bridge copy of the register, and do not generate a read cycle on the PCI bus. For test purposes, this behaviour can be changed by setting bit zero of index register 21h. In this case, a read from IO address 22h reads the South Bridge copy of the register, using a PCI read cycle.

After selecting an index register by writing to IO address 22h, that index register is read from or written to at IO address 23h. Some index registers are implemented in the North Bridge alone, some in the South Bridge alone, and some are duplicated and implemented in both. Whether an index register is implemented in the North Bridge, South Bridge or both is indicated in the description of that register in this document.

For index registers that are implemented in the North Bridge alone, writes to IO address 23h write to the register, and reads of IO address 23h read from the register, and no PCI cycles are generated.

For index registers that are implemented in the South Bridge alone, writes to IO address 23h write to the register, and reads of IO address 23h read from the register. In both cases, the data must go over the PCI bus.

For index registers that are implemented in both the North Bridge and the South Bridge, writes to IO address 23h write to both copies of the register, requiring a PCI write cycle. Reads to IO address 23h reads from the North Bridge copy of the register, and generate no PCI cycles. For test purposes, this behaviour can be changed by setting bit zero of index register 21h. In this case, the South Bridge copy of the register is read, using a PCI read cycle.

### *10.2.3. INTERRUPT ACKNOWLEDGE CYCLE*

When an interrupt is requested, the interrupt controller in the South Bridge asserts the CPU's interrupt input. When the CPU services the interrupt, it must first get the interrupt vector from the interrupt controller. The interrupt vector is used to find the interrupt service routine. Also, since each interrupt request input of the interrupt controller has its own interrupt vector, the vector tells where the interrupt request came from.

To get the interrupt vector, the CPU generates two interrupt acknowledge cycles. Both of these cycles read data from the interrupt controller. The data returned by the first is ignored, while the data for the second contains the interrupt vector in bits 0-7. The North Bridge handles both of the cycles identically, converting them to PCI interrupt acknowledge cycles.

Outside of the interrupt controller, the South Bridge handles both cycles identically. The ISA controller converts the PCI cycles into interrupt acknowledge cycles for the interrupt controllers. The INTA# input of the interrupt controller is asserted for four and a half ISA bus clocks, starting on a falling edge of that clock, and during this time data is transferred from the interrupt controller to the ISA controller. This can be extended to five and a half clocks by setting bit 5 in Index Register 50h.

### *10.2.4. ISA TO PCI READ AND WRITE*

ISA initiated cycles are converted to PCI cycles by the ISA controller. The South Bridge pulls IOCHRDY low to extend these cycles until the PCI cycle has completed.

### *10.2.5. ISA TO PCI BUFFERED READS*

ISA reads of host memory can be buffered. This is disabled by default, and can be enabled by setting bit 6 in Index Register 50h. When this bit is set, ISA bus initiated reads of host memory addresses always get their data from a four byte buffer in the ISA controller which is filled on demand. This can reduce the amount of traffic for a block memory read by up to a factor of four.

The buffer is filled or refilled, under the conditions listed below, after the start of a ISA initiated read of a host memory address has been detected by the South Bridge. The South Bridge generates a PCI read of four bytes, with the low two bits of the address set to zero, and the rest of the address set to be the same as the address on the ISA bus address. The requested data will be driven by the South Bridge onto the ISA bus to finish the ISA read cycle.

The buffer will be refilled if the data requested by the current read is not in the buffer. Also, to avoid stale data, the buffer will be refilled for:

- The first host memory read after an ISA bus master gets ownership of the bus,
- The first host memory read after any ISA bus cycle which is not a host memory read,
- Any ISA read of a byte in the buffer which has already been read since the buffer was last filled.

If a host memory read can be fulfilled without refilling the buffer, no PCI cycle is generated.

### *10.2.6. ISA TO PCI POSTED WRITES*

ISA writes to host memory can be posted. This is disabled by default, and can be enabled by setting bit 7 in Index Register 50h. When this bit is set, ISA bus initiated writes to host memory addresses go to a four byte buffer in the ISA controller. No PCI write is generated until the buffer is written to host memory.

The buffer is written to host memory when:

- The buffer gets full,
- or there is a host memory write to a location not in the buffer,
- or a host memory write would overwrite data already in the buffer,
- or there is an ISA cycle which is not a host memory write,
- or the current ISA master gives up ownership of the bus.

If writing the buffer to host memory is triggered by an ISA bus cycle, that cycle is held up by pulling IOCHRDY low until the buffer has been written to host memory.

Note that it is possible for the South Bridge to generate writes with discontiguous byte enables if posted writes are enabled.

### *10.2.7. ISA TO REGISTER READ AND WRITE*

ISA initiated cycles which target South Bridge internal registers will first be tried on the PCI bus. If they are not claimed by a PCI target, then the register will be read or written. Reads and writes to IPC registers will cause the South Bridge to pull IOCHRDY low for at least the number of cycles programmed into Index Register 01h. Reads and writes to the South Bridge registers which are not IPC registers are normally

disabled. These can be enabled by setting bit 7 of Index Register 51h. Writes to these registers require a longer than standard recovery time of two ISACLK periods.

### 10.3. XBUS READ AND WRITE

The XBUS is an 8-bit subset of the ISA bus that connects low speed devices on the mother board to the CPU. In particular, the Real Time Clock (RTC), the Keyboard Controller, and the BIOS ROM will usually be connected via the XBUS. For the STPC, the XBUS shares address, data and command lines with the ISA bus. No buffers or transceivers are required to connect the XBUS to the ISA bus. The timing for XBUS cycles is the same as that for eight bit ISA cycles, see above.

#### *10.3.1. REAL TIME CLOCK READ AND WRITE*

The Real Time Clock (RTC) is connected to the XBUS. However the RTC is not connected to the command lines of the XBUS. Instead, four input pins of the RTC (CS#, AS, RW#, DS) are controlled directly by the STPC. The MOT pin of the RTC must be tied low. The registers in the RTC are accessed indirectly, by first writing the register number to IO port 70h, and then reading or writing the register at IO port 71h.

The RTC input CS# is connected to the logical OR of the outputs RMRTCCS# and ISAOE#. CS# is the chip select for the RTC, and it will be driven low (active) on any IO read or write to port 70h or port 71h, and also will be driven low by reads or writes to ROM address space.

The RTC input AS is directly connected to the RTCAS output. AS is the address strobe for the RTC, and it is asserted (high) during any IO write to port 70h.

The RTC input RW# is connected to the logical OR of the RTCRW# and ISAOE# outputs. RW# is write pulse for the RTC, and it will be asserted (low) during any IO write to port 71h.

The RTC input DS is connected to the logical OR of the South Bridge outputs RTCDS and ISAOE#. DS is the read pulse for the RTC, and it will be asserted (low) during any IO read of port 71h.

The RTC interrupt output IRQ# is directly connected to the IRQ8 input. There is an internal inverter between the pin IRQ8 and the interrupt controller to maintain compatibility with the PC-AT without requiring additional external glue logic.

#### *10.3.2. KEYBOARD CONTROLLER READ AND WRITE*

The keyboard controller is connected to the XBUS. The chip select input of the keyboard controller is connected to the logical OR of the KBCS# and ISAOE# outputs.

Reads and writes to IO addresses 60h, 62h, 64h, 66h, 68h, 6Ah, 6Ch, and 6Eh are taken by the South Bridge to be reads and writes to the keyboard controller. Writes to the keyboard controller may be intercepted by South Bridge for keyboard controller emulation. In this case, neither IOW# or KBCS# will be asserted. For writes to the keyboard controller that are not intercepted, both IOW# and KBCS# will be asserted (low) during the write. Similarly, for any reads from the keyboard controller, both IOR# and KBCS# will be asserted (low) during the read.

#### *10.3.3. BIOS ROM READ AND WRITE*

The BIOS ROM is connected to the XBUS. The chip select for the ROM is connected to the logical OR of the RMRTCCS# and ISAOE# outputs.

### *10.3.4. CPU RESET AND GATE A20*

Before the 286 CPU, memory space was limited to 1MB. Some software applications used this characteristic to access data in segment 0 by generating an address above the 1MB. To stay compatible with these applications, when the 286 appeared, the PC motherboards included, via the keyboard controller, a mechanism to enable or disable this pre-286 compatibility. This is done by the 'Gate A20' mechanism. When enabled, the CPU A20 address line is propagated to the memory bus. When disabled, the memory bus A20 line is forced to 0 (8086 compatible).

To be able to reset the CPU, the keyboard controller also includes a pin which is connected to the CPU reset pin (only the CPU is reset, not the chipset or external components).

The STPC doesn't provide external pins to be able to control the gate A20 and CPU reset. These features are controlled internally by the keyboard emulation. Thus, the STPC checks the commands and data sent to the keyboard controller, and when it recognizes the related commands, applies them internally. The keyboard emulation must be on (register STPC_MISC0/bit 3 = 0), else you won't be able to reset the CPU and the A20 line will always be masked. This is done in the default configuration.

Notes; Only the P2 write command (D1h) and the reset pulse command (FEh) are emulated. In particular, the P2 read command (D0h) is not emulated, so the return value is the keyboard controller P2 state.

On the other STPCs, the commands and data are forwarded, so the keyboard controller receives and applies them, the resulting actions are ignored by the STPC, but the keyboard controller A20 state reflects the STPC A20 state.

#### *10.3.4.1. Reset Method*

To disable Gate A20 (forcing address bit 20 to low), write D1h to the I/O Port 64h then write xxxx xx0xb to I/O Port 60h.
To enable Gate A20 (forcing address bit 20h to high

The Reset, also known as warm reset, is generated by writing data FEh to I/O port 64h or by writing data DEh to I/O port 64h then writing data xxxxxxx0 binary (bit 0 = ‘0’) to I/O port 60h.

Fast host CPU reset only is generated by two methods:

1) Whenever the STPC detects a write to Port 64h with data FEh.

2) Whenever the STPC detects a write to Port 60h following a D1h data write to Port 64h, bit 0 of the data byte being written at Port 60h is ‘0’.

The CPU reset is at least 16 host clocks. The write cycle is not forwarded to the keyboard controller.

## 10.4. ISA STANDARD REGISTERS

The ISA standard registers correspond to the registers in the peripheral components integrated in the STPC as well as the miscellaneous ports implemented on a ISA motherboard. These registers reside in IO space.

The functions controlled by the ISA registers include the DMA and interrupt control, BIOS and keyboard interface.

### *10.4.1. DMA 1 CONTROLLER REGISTERS*

DMA 1 controls 8-bit DMA transfers.

There are 16 DMA 1 registers. They are as shown in Table 10-1.

**Table 10-1. DMA1 Registers**

| IO address bits 15-0 | Reset Value | Register Name | Mnemonic |
|---|---|---|---|
| XXXX XX00 000x 0000 | xxxx xxxx | DMA 1 Channel 0 Base and Current Address | DMA1_CBA0 |
| XXXX XX00 000x 0001 | xxxx xxxx | DMA 1 Channel 0 Base and Current Count | DMA1_CBC0 |
| XXXX XX00 000x 0010 | xxxx xxxx | DMA 1 Channel 1 Base and Current Address | DMA1_CBA1 |
| XXXX XX00 000x 0011 | xxxx xxxx | DMA 1 Channel 1 Base and Current Count | DMA1_CBC1 |
| XXXX XX00 000x 0100 | xxxx xxxx | DMA 1 Channel 2 Base and Current Address | DMA1_CBA2 |
| XXXX XX00 000x 0101 | xxxx xxxx | DMA 1 Channel 2 Base and Current Count | DMA1_CBC2 |
| XXXX XX00 000x 0110 | xxxx xxxx | DMA 1 Channel 3 Base and Current Address | DMA1_CBA3 |
| XXXX XX00 000x 0111 | xxxx xxxx | DMA 1 Channel 3 Base and Current Count | DMA1_CBC3 |
| XXXX XX00 000x 1000 | xxxx 0000 | DMA 1 Read Status/Write Command register | DMA1_RSWC |
| XXXX XX00 000x 1001 | 1111 xxxx | DMA 1 Request register | DMA1_RR |
| XXXX XX00 000x 1010 | 0000 0000 | DMA 1 Read Command/Write Single Mask register | DMA1_RCWSM |
| XXXX XX00 000x 1011 | 0000 0000 | DMA 1 Mode register | DMA1_Mode |
| XXXX XX00 000x 1100 | 1111 1111 | DMA 1 Set/Clear Byte pointer flip-flop | DMA1_SCBPFF |
| XXXX XX00 000x 1101 | 0000 0000 | DMA 1 Read Temp register/Master Clear | DMA1_RTMC |
| XXXX XX00 000x 1110 | 1111 1111 | DMA 1 Clear Mask/Clear all request | DMA1_CMCAR |
| XXXX XX00 000x 1111 | 1111 1111 | DMA 1 Read/Write all Mask register bits | DMA1_RWMB |

Note that the not all bits of the address are used.

Programming notes:

Channel 0 corresponds to the internal DRQ0B, channel 1 to DRQ1B, channel 2 to DRQ2B, and channel 3 corresponds to the internal DRQ3B.

### *10.4.2. INTERRUPT CONTROLLER 1 REGISTERS*

There are two interrupt controller 1 registers. They are as shown in Table 10-2.

Interrupt controller 1 is the master interrupt controller.

**Table 10-2. Interrupt Controller 1 Registers**

| IO address bits 15-0 | Reset Value | Register Name | Mnemonic |
|---|---|---|---|
| 0000 0000 0010 0000 | 0000 0000 | Interrupt Controller 1 register | IC_1 |
| 0000 0000 0010 0001 | 1111 1111 | Interrupt Controller 1 Mask register | IC_1MR |

Note that not all bits of the address are used.

Programming notes:

Interrupt controller 1 input IR0 is connected IRQ0, IR1 to IRQ1, IR2 to interrupt out from interrupt controller 2, IR3 to IRQ3, IR4 to IRQ4, IR5 to IRQ5, IR6 to IRQ6, and IR7 to IRQ7.

### *10.4.3. INTERVAL TIMER REGISTERS*

The Interval Timer comprises three independent counters. Counter 0 is used to generate timer interrupts, counter 1 is used to generate ISA bus refresh, and counter 2 is used to create the speaker tone.

There are 4 Interval Timer registers. They are as shown in Table 10-3.

**Table 10-3. Interval Timer Registers**

| IO address bits 15-0 | Reset Value | Register Name | Mnemonic |
|---|---|---|---|
| XXXX XX00 010x xx00 | xxxx xxxx | Interval Timer Register Counter 0 Count | IT_0 |
| XXXX XX00 010x xx01 | xxxx xxxx | Interval Timer Register Counter 1 Count | IT_1 |
| XXXX XX00 010x xx10 | xxxx xxxx | Interval Timer Register Counter 2 Count | IT_2 |
| XXXX XX00 010x xx11 | 1111 1111 | Command Mode register | IT_3 |

Note that not all bits of the address are decoded.

Programming notes:

All three counters are clocked by 1.193 MHz nominal frequency (OSC/12). Counter 0 and counter 1 gates are always on, counter 2 gate is controlled by writing to Port B (see Section 10.4.4.).

### *10.4.4. PORT B REGISTER*

This is the ISA compatible 8-bit Port B register located at xxxx xxxx 0110 xxx1 IO address (bits 15-0). It has the following meaning:

*Port_B* Access = 0061h Regoffset =

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PE | IOCHK | T/C 2S | ISA RC | ISA IOCHK | PCE | SE | T/C 2 G |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | PE | **Parity Error.** This bit is set to a '1' whenever a parity error is detected during system memory read operation. Once set, this bit can be cleared by setting bit 2 of this register to a '1'. Bit 2 should be reset to a '0' to enable recording the next parity error. The parity error generates NMI to the host CPU if NMI is enabled.This bit is read-only. |
| Bit 6 | IOCHK | ISA IOCHK# Enable. This bit is set to a '1' when IOCHK# signal of the ISA bus is asserted. Once set, this bit is cleared by setting bit 3 of this register to a '1'. Bit 3 should be reset to a '0' to enable recording the next IOCHK#. IOCHK# generates NMI to the host CPU if NMI is enabled. This bit is read only. |
| Bit 5 | T/C 2S | ISA T/C 2 State. This bit reflects the output of Timer/Counter 2 without any synchronization. This bit is read only. |
| Bit 4 | ISA RC | ISA Refresh Check. This bit toggles on every rising edge of the REFRESH# signal of the ISA bus. This bit is read only. |
| Bit 3 | ISA IOCHK | ISA IOCHK# Enable. This bit is connected to the asynchronous clear input of the flipflop which records the IOCHK#. It must be set to a '1' to clear the flipflop and then set to a '0' to enable further IOCHK# assertions. This bit is read/write and cleared to a '0' by ISA reset. |
| Bit 2 | PCE | **Parity Check Enable.** This bit is connected to the asynchronous clear input of the flipflop which records the parity error. It must be set to a '1' to clear the flipflop and then set to a '0' to enable further parity errors. This bit is read/write and cleared to a '0' by ISA reset. |
| Bit 1 | SE | ISA Speaker Enable. This bit is ANDed with the Interval Timer counter 2 OUT signal to drive the Speaker output signal. This bit is read/write and cleared to a '0' by ISA reset. |
| Bit 0 | T/C 2G | T/C 2 Gate. This bit is connected to the gate input of the Interval Timer counter 2. This bit is read/write and cleared to a '0' by ISA reset. |

### *10.4.5. PORT 70H REGISTER*

This 8-bit write-only register contains the NMI enable bit and is located at xxxx xxxx 0111 0xx1 IO address.

*Port_70* Access = 0070h Regoffset =

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| NMI E | Rsv | | | | | | |
| Default value after reset = 80h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | NMI E | NMI Enable. NMI is asserted on encountering IOCHK# on the ISA bus (Port_B) or SERR# on the PCI bus if this bit is set to a '0'. Setting this bit to a '1' disables NMI generation. |
| Bit 6-0 | Rsv | **Reserved.** must be written to '0's. Read back is undefined. |

Programming notes:

Writing to this address also sets the address register in the Real Time Clock (RTC, not part of the STPC, normally connected via the ISA interface).

### *10.4.6. INTERRUPT CONTROLLER 2 REGISTERS*

Interrupt controller 2 is the slave interrupt controller.

Interrupt controller 2 occupies two register locations. They are as shown in Table 10-4.

**Table 10-4. Interrupt Controller 2 Registers**

| IO address bits 15-0 | Reset Value | Register Name | Mnemonic |
|---|---|---|---|
| XXXX XX00 101x xxx0 | 0000 0000 | Interrupt Controller 2 register | IC_2R |
| XXXX XX00 101x xxx1 | 1111 1111 | Interrupt Controller 2 Mask register | IC_2M |

Note that not all address bits are decoded.

Programming notes:

Interrupt controller 2 input IR1 is connected to IRQ9, IR2 to IRQ10, IR3 to IRQ11, IR4 to IRQ12, IR6 to IRQ14, IR7 to IRQ15. IR0 driven by IRQ8 inverted. IR5 is driven by an internally generated floating point error interrupt request.

### *10.4.7. DMA CONTROLLER 2 REGISTERS*

There are 16 DMA 2 registers. They are as shown in Table 10-5.

**Table 10-5. DMA Controller 2 Registers**

| IO address bits 15-0 | Reset Value | Register Name | Mnemonic |
|---|---|---|---|
| XXXX XX00 1100 000x | xxxx xxxx | DMA 2 Channel 0 Base and Current Address | DMA2_CBAO |
| XXXX XX00 1100 001x | xxxx xxxx | DMA 2 Channel 0 Base and Current Count | DMA2_CBCO |
| XXXX XX00 1100 010x | xxxx xxxx | DMA 2 Channel 1 Base and Current Address | DMA2_CBA1 |
| XXXX XX00 1100 011x | xxxx xxxx | DMA 2 Channel 1 Base and Current | DMA2_CBC1 |
| XXXX XX00 1100 100x | xxxx xxxx | DMA 2 Channel 2 Base and Current Address | DMA2_CBA2 |
| XXXX XX00 1100 101x | xxxx xxxx | DMA 2 Channel 2 Base and Current | DMA2_CBC2 |
| XXXX XX00 1100 110x | xxxx xxxx | DMA 2 Channel 3 Base and Current Address | DMA2_CBA3 |
| XXXX XX00 1100 111x | xxxx xxxx | DMA 2 Channel 3 Base and Current Count | DMA2_CBC3 |
| XXXX XX00 1101 000x | 1111 xxxx | DMA 2 Read Status/Write Command register | DMA2_RSWC |
| XXXX XX00 1101 001x | 0000 0000 | DMA 2 Request register | DMA2_RR |
| XXXX XX00 1101 010x | 0000 0000 | DMA 2 Read Command/Write Single Mask register | DMA2_RCWSM |
| XXXX XX00 1101 011x | 0000 0000 | DMA 2 Mode register | DMA2_Mode |
| XXXX XX00 1101 100x | 1111 1111 | DMA 2 Set/Clear Byte pointer flip-flop | DMA2_SCBPFF |
| XXXX XX00 1101 101x | 0000 0000 | DMA 2 Read Temporary/Master Clear | DMA2_RTMC |
| XXXX XX00 1101 110x | 1111 1111 | DMA 2 Clear Mask/Clear all requests register | DMA2_CMCAR |
| XXXX XX00 1101 111x | 1111 1111 | DMA 2 Read/Write all Mask register bits | DMA2_RWMRB |

Note that the not all bits of the address are used.

### *10.4.8. DMA PAGE REGISTERS*

The DMA Page registers defines address bits [16-23] for DMA transfers controlled by DMA 1 or DMA 2. Bits [0-15] are generated by the DMA controller, bits [16-23] come from the appropriate page register, and bits 31-24 are all zeroes.

There are 16 DMA page registers. They are as shown in Table 10-6.

**Table 10-6. DMA Page Registers**

| IO address bits 15-0 | Reset Value | Register Name | Mnemonic |
|---|---|---|---|
| XXXX XX00 100x0000 | xxxx xxxx | DMA Page Register Port 80h (Reserved) | Port_80 |
| XXXX XX00 100x0001 | xxxx xxxx | DMA Page Register Channel 2 | DMA_PRC2 |
| XXXX XX00 100x0010 | xxxx xxxx | DMA Page Register Channel 3 | DMA_PRC3 |
| XXXX XX00 100x0011 | xxxx xxxx | DMA Page Register Channel 1 | DMA_PRC1 |
| XXXX XX00 100x0100 | xxxx xxxx | DMA Page Register Port 84h (Reserved) | Port_84 |
| XXXX XX00 100x0101 | xxxx xxxx | DMA Page Register Port 85h (Reserved) | Port_85 |
| XXXX XX00 100x0110 | xxxx xxxx | DMA Page Register Port 86h (Reserved) | Port_86 |
| XXXX XX00 100x0111 | xxxx xxxx | DMA Page Register Channel 0 | DMA_PRC0 |
| XXXX XX00 100x1000 | xxxx xxxx | DMA Page Register Port 87h (Reserved) | Port_87 |
| XXXX XX00 100x1001 | xxxx xxxx | DMA Page Register Channel 6 | DMA_PRC6 |
| XXXX XX00 100x1010 | xxxx xxxx | DMA Page Register Channel 7 | DMA_PRC7 |
| XXXX XX00 100x1011 | xxxx xxxx | DMA Page Register Channel 5 | DMA_PRC5 |

**Table 10-6. DMA Page Registers**

| | | | |
|---|---|---|---|
| XXXX XX00 1000 1100 | xxxx xxxx | DMA Page Register Port 8Bh (Reserved) | Port_8B |
| XXXX XX00 1000 1101 | xxxx xxxx | DMA Page Register Port 8Ch (Reserved) | Port_8C |
| XXXX XX00 1000 1110 | xxxx xxxx | DMA Page Register Port 8Dh (Reserved) | Port_8D |
| XXXX XX00 1000 1111 | xxxx xxxx | DMA Page Register Port 8Eh (Reserved) | Port_8E |

## 10.5. ISA CONFIGURATION REGISTERS

These registers are addressed through the Address Configuration Index (CI) and Data registers.

### *10.5.1. MISCELLANEOUS CONTROL REGISTER 0*

*Misc_Cont0* Access = 0022h/0023h Regoffset = 050h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ISA WPE | ISA RBE | ISA WIC | ISA CFS | KRE | CPU D | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | ISA WPE | **ISA Write Post Enable**. If '1', posted writes to host memory by ISA DMA or ISA bus master are enabled. |
| Bit 6 | ISA RBE | **ISA Read Buffer Enable.** If '1', buffered reads of host memory by ISA DMA or ISA bus master are enabled. |
| Bit 5 | ISA WIC | **ISA Wait Insert Control.** This bit controls if extra wait state is inserted for slower ISA devices.<br>0: No extra wait state for ISA cycle<br>1: One extra wait state for ISA cycle |
| Bit 4 | ISA CFS | **ISA Clock Frequency Select.** This bit selects the ISA clock frequency.<br>0: ISA clock is 14.31818 MHz / 2<br>1: ISA clock is PCICLK /4 |
| Bit 3 | KRE | **Keyboard Reset Enable.** This bit if set to a '1', keyboard emulation fast gate A20 and fast reset are disabled. The source of warm reset indication is from the keyboard controller and the CPU core will use the gate A20 indication from keyboard controller for its internal A20M# input. |
| Bits 2-0 | CPU D | **CPU Deturbo.** These three bits define the ratio CPU is held. (see Table 10-7). |

**Table 10-7. CPU Deturbo**

| Bit 2 | Bit 1 | Bit 0 | CPU Deturbo |
|---|---|---|---|
| 0 | 0 | 0 | deturbo is disabled. |
| 0 | 0 | 1 | CPU is held 1/2 of the time. |
| 0 | 1 | 0 | CPU is held 2/3 of the time. |
| 0 | 1 | 1 | CPU is held 3/4 of the time. |
| 1 | 0 | 0 | CPU is held 4/5 of the time. |
| 1 | 0 | 1 | CPU is held 5/6 of the time. |
| 1 | 1 | 0 | CPU is held 6/7 of the time. |
| 1 | 1 | 1 | CPU is held 7/8 of the time. |

### *10.5.2. MISCELLANEOUS CONTROL REGISTER 1*

*Misc_Cont1* Access = 0022h/0023h Regoffset = 051h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IPC W | CLK 24 | HCLK D | Rsv | ROM | S E S | S D S | S C S |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 7 | IPC W | **IPC Write control.** This bit controls the ISA master writes to the IPC register<br>0: ISA master writes to IPC register disabled<br>1: ISA master writes to IPC register enabled |
| Bit 6 | CLK 24 | **CLK24 Disable.** This bit controls the output of CLK24.<br>0: CLK24 generated normally<br>1: Clock synthesiser for CLK24 is disable (CLK24 will not toggle) |
| Bit 5 | HCLK D | **HCLK Disable.** This bit controls the generation of HCLK.<br>0: HCLK generated normally<br>1: Clock synthesiser for HCLK is disabled (HCLK will not toggle) |
| Bit 4 | Rsv | **Reserved**. |
| Bit 3 | ROM | **ROM Write Protect Enable.** This bit, if set to a '1', disables write cycles to ROM BIOS on extended bus. If set to '0', write to extended bus ROM BIOS is allowed.<br>Note: This bit can not disable the write to shadowed BIOS in DRAM since after shadow is enabled, all writes to BIOS should be forwarded to extended bus. |
| Bit 2 | S E S | **Segment E Share.** This bit controls if E0000h-EFFFFh segment shares the FLASH memory with F0000h-FFFFFh segment.<br>0: Sharing disabled<br>1: Sharing enabled |
| Bit 1 | S D S | **Segment D Share.** This bit controls if D0000h-DFFFFh segment shares the FLASH memory with F0000h-FFFFFh segment.<br>0: Sharing disabled<br>1: Sharing enabled |
| Bit 0 | S C S | **Segment C Share.** This bit controls if C0000h-CFFFFh segment shares the FLASH memory with F0000h-FFFFFh segment.<br>0: Sharing disabled<br>1: Sharing enabled |

### *10.5.3. PIRQ ROUTING CONTROL REGISTER 0*

This 8-bit register controls the routing of PCI Interrupt A# to one of the interrupt inputs of the 8259 as follows. It applies to interrupts A# to D#:

| PIRQ Routing | Regoffset |
|---|---|
| PIRQ A | 52h |
| PIRQ B | 53h |
| PIRQ C | 54h |
| PIRQ D | 55h |

*PAR_Cont0* Access = 0022h/0023h Regoffset = Table 8-1.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RE | Rsv | | | RC A | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **RE** | **Routing Enable.** If set to a '1', this bit enables the routing of PCI interrupt, otherwise the PCI interrupt A# is unconnected. |
| **Bits 6-4** | **Rsv** | **Reserved**. Writes have no affect. Reads return undefined value. |
| **Bits 3-0** | **RC A** | **Routing Control.** These bits route the PCI interrupt A# (see Table 10-8) |

**Table 10-8. Routing Control Encoding**

| Bit 3 | Bit 2 | Bit 1 | Bit 0 | Interrupt A# Route | Note |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | Reserved | 1 |
| 0 | 0 | 0 | 1 | Reserved | 1 |
| 0 | 0 | 1 | 0 | Reserved | 1 |
| 0 | 0 | 1 | 1 | IRQ3 | |
| 0 | 1 | 0 | 0 | IRQ4 | |
| 0 | 1 | 0 | 1 | IRQ5 | |
| 0 | 1 | 1 | 0 | IRQ6 | |
| 0 | 1 | 1 | 1 | IRQ7 | |
| 1 | 0 | 0 | 0 | Reserved. | 1 |
| 1 | 0 | 0 | 1 | IRQ9 | |
| 1 | 0 | 1 | 0 | IRQ10 | |
| 1 | 0 | 1 | 1 | IRQ11 | |
| 1 | 1 | 0 | 0 | IRQ12 | |
| 1 | 1 | 0 | 1 | Reserved. | 1 |
| 1 | 1 | 1 | 0 | IRQ14 | |
| 1 | 1 | 1 | 1 | IRQ15 | |

Note 1: Interrupt can not be routed to this level.

### *10.5.4. INTERRUPT LEVEL CONTROL REGISTER 0*

This 8-bit register allows interrupt requests to the lower 8259 to be either level or edge sensitive on an interrupt by interrupt basis overriding the global edge/level control bit of the 8259.

*IRQ_Lev_C_0* Access = 0022h/0023h Regoffset = 056h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IRQ C | | | | | Rsv | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-3** | **IRQ C** | **IRQ Control IRQ[7-3].** If set to a '1', the corresponding interrupt request is treated as a level input, otherwise it is treated as the edge sensitive input (compatible to ISA). |
| **Bits 2-0** | **Rsv** | **Reserved.** Writes have no affect. Reads return undefined value. |

### *10.5.5. INTERRUPT LEVEL CONTROL REGISTER 1*

This register allows interrupt requests to the upper 8259 to be either level or edge sensitive on an interrupt by interrupt basis overriding the global edge/level control bit of the 8259. This register has the following definition.

*IRQ_Lev_C_1* Access = 0022h/0023h Regoffset = 057h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IRQ C | | Rsv | IRQ C | | | | IPC |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-6** | **IRQ C** | **IRQ Control IRQ[15-14].** If set to a '1', the corresponding interrupt request is treated as a level input, otherwise it is treated as the edge sensitive input (compatible with ISA). |
| **Bit 5** | **Rsv** | **Reserved.** Writes have no affect and the reads return undefined value. |
| **Bits 4-1** | **IRQ C** | **IRQ Control IRQ[12-9].** If set to a '1', the corresponding interrupt request is treated as a level input, otherwise it is treated as the edge sensitive input (compatible to ISA). |
| **Bit 0** | **IPC** | This bit controls the ISA refresh cycle.Setting to 0 disables ISA refresh and setting to 1 enables ISA refresh.<br>By setting this bit to 1 enables the toggling the ISA Port B refresh bit (see Section 10.4.4. ). |

### *10.5.6. IPC CONFIGURATION REGISTER*

This 8-bit register controls the timing of the DMA controllers, and also the number of wait states for writes to registers in the IPC.

*IPC_Conf* Access = 0022h/0023h Regoffset = 001h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IPC WS | | DMA | | DMA | | DMA M | DMA C |
| Default value after reset = C0h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-6** | **IPC WS** | **IPC Wait States.** These bits specify the number of ISACLK wait states for read or write to IPC register1 (see Table 10-9). |
| **Bits 5-4** | **DMA** | **DMA 16-Bit Wait States.** These bits specify the number of wait states in 16-bit DMA cycles (see Table 10-10). |
| **Bits 3-2** | **DMA** | **DMA 8-Bit Wait States.** These bits specify the number of wait states in 8 bit DMA cycle (see Table 10-11). |
| **Bit 1** | **DMA M** | **DMA MEMR# Timing.** If this bit is set to '1' the DMA controllers will assert MEMR# at the the same time as IOW#. If set to '0' (default), MEMR# will be asserted one clock after IOW#. |
| **Bit 0** | **DMA C** | **DMA Clock Select.** If this bit is set to '0' (default), the DMA controller clock will be ISACLK divided by two, otherwise the DMA controller clock will be ISACLK. |

**Table 10-9. IPC Wait States**

| Bit 7 | Bit 6 | IPC Wait States |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 2 |
| 1 | 0 | 3 |
| 1 | 1 | 4 (Default) |

**Table 10-10. DMA 16-bit Wait States**

| Bit 5 | Bit 4 | DMA 16-bit Wait States |
|---|---|---|
| 0 | 0 | 1 (Default) |
| 0 | 1 | 2 |
| 1 | 0 | 3 |
| 1 | 1 | 4 |

**Table 10-11. DMA 8-bit Wait States**

| Bit 3 | Bit 2 | DMA 8-bit Wait States |
|---|---|---|
| 0 | 0 | 1 (Default) |
| 0 | 1 | 2 |
| 1 | 0 | 3 |
| 1 | 1 | 4 |

Programming notes:

To read or write to this register, write 01 to index register 22h, and then read or write from data register 23h.

### *10.5.7. ISA SYNCHRONISER BYPASS REGISTER*

This 8-bit register controls whether or not the signals between the PCI logic and the ISA logic are passed through synchronization logic. This bit would normally be set only when the ISA clock is derived from PCI clock (that is index 50h, bit 4 is set to '1'). Setting this bit will result in a small improvement in ISA performance.

*ISA_Sync* Access = 0022h/0023h Regoffset = 059h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | SE |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-1** | **Rsv** | **Reserved.** Writes have no affect. Reads return undefined value. |
| **Bit 0** | **SE** | **Synchronization Enable.**<br>**0: Enabled**<br>**1: Disabled** |

# 11. IDE CONTROLLER

## 11.1. INTRODUCTION

The IDE (Integrated Drive Electronics) controller provides two IDE channels, primary and secondary, for interfacing with up to four IDE drives. It supports PIO modes 0 to 4 plus DMA modes 0 to 2. The timings are individually programmable for all four IDE devices. Each channel has a four double-word FIFO for data transfers which allows four levels of write posting or read prefetch. Accesses to the 8-bit non-data IDE registers bypass the FIFOs.

For each of the four drives there are three bits in the configuration registers which can selectively enable write posting, read prefetch and ATAPI read prefetch. If read prefetch is enabled, the IDE controller will prefetch data from the drive after the first read has been made. The prefetching will stop after 256 data reads (512 Bytes), which is the normal sector size. If the current command to the drive is ATAPI packet (A0h), or service (A2h), then the read prefetch will be disabled unless ATAPI read prefetch is set.

The two channels of the IDE controller can be individually programmed to operate in either legacy or native mode. In legacy mode, the IDE interrupts are hardwired to INT 14 & 15. In native mode, they both connect to PCI INTA. If legacy mode is selected, INT 14 & 15 will not be available on the ISA bus even if IDE interrupts are disabled. In legacy mode, the primary and secondary channels are hardwired to IO addresses 1F0h-1F7h / 3F6h and 170h-177h / 376h respectively. In native mode the IO addresses are programmed by configuration registers. For information on PIO mode, please refer to the ATAPI Standard.

The IDE controller provides DMA bus master transfer between IDE devices and system memory, with scatter/gather capability. By performing the IDE data transfer as a bus master, the Bus Master Device off-loads the CPU (no programmed IO for data transfer) and improves system performance in multitasking environments.

Before issuing the DMA command, the system software must first create a Physical Region Descriptor (PRD) table in system memory. This table contains a list of pointers and byte counts for each entry. A register in the IDE controller is set to point to this table. The IDE DMA controller will read from system memory during DMA initialization. Each entry in the PRD table is eight bytes long and will have the format below:

### 11.2. PRD TABLE ENTRY

*PRD1*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EOT | Rsv | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NOW | | | | | | | | | | | | | | | Rsv |
| Default value after reset = undefined | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 31** | **OET** | This bit set to '1' if it is the last entry in the table |
| **Bits 30-16** | **Rsv** | **Reserved** |
| **Bit 15-1** | **NOWS** | **Number of 16-bit data packets** |
| **Bit 0** | **Rsv** | **Reserved;** This bit must be set to 0 |

*PRD0*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MRPAS | | | | | | | | | | | | | | | |
| Default value after reset = undefined | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MRPAS | | | | | | | | | | | | | | | Rsv |
| Default value after reset = undefined | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 31-1** | **MRPAS** | **Memory Physical Address of the first descriptor** |
| **Bit0** | **Rsv** | **Reserved;** This bit must be set to 0 |

The table must be aligned on a 4 byte boundary and should not cross a 64k boundary.

A memory region also should not cross a 64k boundary neither. An example of a PRD table is shown in Figure 11-1..

The primary and secondary channels each have a PRD address pointer register.

To save pins, the IDE controller shares pins with the ISA interface. On the IDE data bus, CS1 & CS3 signals are shared with the ISA address bus and keyboard controller/RTC pins. These signals are isolated by external transceiver devices. The ISAOE signal selects whether the pins are in IDE or ISA mode. The

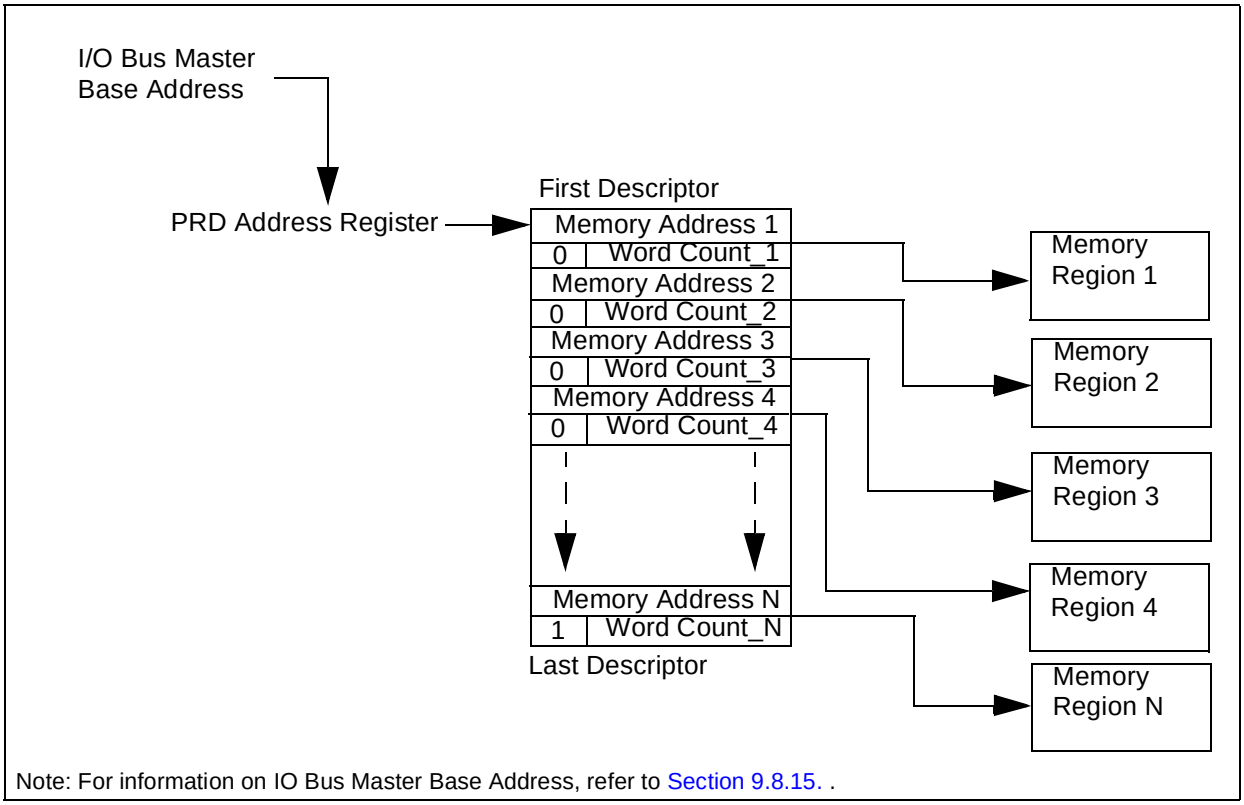



**Figure 11-1. PRD Table Entry Example**

South Bridge arbitrates between the IDE controller and the ISA bus bridge to select which has control of the shared pins.

## 11.3. IDE BUS MASTER REGISTERS

This document defines a register level programming interface for the internal busmaster ATA-compatible (IDE) disk controller that directly moves data between IDE devices and main memory.

The system using this programming interface will benefit from bundled software shipped with major Operating Systems, limiting the amount of software development required to provide a complete product.

The master mode programming interface is an extension of the standard IDE programming model. This means that devices can always be dealt with using the standard IDE programming model, with the master mode functionality used when the appropriate driver and devices are present. Master operation is designed to work with any IDE device that supports DMA transfers on the IDE bus. Devices that only work in PIO mode can be used through the standard IDE programming model.

The programming interface defines a simple scatter/gather mechanism, allowing large transfer blocks to be scattered to or gathered from memory. This cuts down the number of interrupts to and interactions with the CPU. The interface defined here supports two IDE channels (primary and secondary). Individual controllers that support more than two channels will need to appear to software as multiple controllers if the standard drivers are to be used. Master IDE controllers should default to Mode 0 Multiword DMA timings to ensure operation with DMA capable IDE devices without the need for controller-specific code to initialize controller-specific timing parameters.

### *11.3.1. PHYSICAL REGION DESCRIPTOR TABLE*

Before the controller starts a master transfer, it is given a pointer to a Physical Region Descriptor Table. This table contains some of a number of the Physical Region Descriptors (PRD); these define the memory areas that are involved in the data transfer. The descriptor table must be aligned on a 4 Byte boundary and the table cannot cross a 64 KByte boundary in memory.

### *11.3.2. PHYSICAL REGION DESCRIPTOR*

The physical memory region to be transferred is described by a Physical Region Descriptor (PRD). The data transfer will not proceed until all regions described by the PRDs in the table have been transferred.

Each Physical Region Descriptor entry is eight Bytes long.

- The first four bytes specify the byte address of a physical memory region.
- The next two bytes specify the count of the region in bytes (64K byte limit per region).

A value of zero in these two bytes indicates 64 KByte. Bit 7 of the last byte indicates the end of the table; the Bus Master operation terminates when the last descriptor has been retired.

**Note:** The memory region specified by the descriptor is further restricted such that the region cannot straddle a 64K boundary. This means that the byte count can be limited to 64K, and the incrementer for the current address register need only extend from bit [1] to bit [15]. Also, the total sum of the descriptor byte counts must be equal to, or greater than, the size of the disk transfer request. If greater than, then the driver must terminate the Bus Master transaction (by resetting bit zero of the command register to zero) when the drive issues an interrupt to signal transfer completion.

## 11.4. BUS MASTER IDE REGISTER DESCRIPTION

The bus master IDE function uses 16 bytes of IO space. All bus master IDE IO space registers can be accessed as byte, word or Dword quantities. The description of the 16 bytes of IO registers is given in Table 11-5.

**Table 11-5. Bus Master IDE Register Description**

| Offset | Register | R/W Status |
|---|---|---|
| 00h | Bus Master IDE Command register Primary | R/W |
| 01h | Device Specific | |
| 02h | Bus Master IDE Status register Primary | RWC |
| 03h | Device Specific | |
| 04h-07h | Bus Master IDE PRD Table Address Primary | R/W |
| 08h | Bus Master IDE Command register Secondary | R/W |
| 09h | Device Specific | |
| 0Ah | Bus Master IDE Status register Secondary | RWC |
| 0Bh | Device Specific | |
| 0Ch-0Fh | Bus Master IDE PRD Table Address Secondary | R/W |

## 11.6. BUS MASTER IDE COMMAND REGISTER

### *11.6.1. IDE COMMAND REGISTER*

This 8-bit Register is addressed at offset Base + 00h for the Primary IDE Channel and Base + 08h for the Secondary IDE Channel.

This register enables/disables Bus Master capability for the IDE function and provides direction control for the IDE DMA transfers. This register also provides the bits that software uses to indicate DMA capability of the IDE device.

*IDE_COM*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | | | RWCOM | Rsv | | SSBM |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7-4** | **Rsv** | **Reserved** |
| **Bit 3** | **RWCOM** | **Read Write Control.** This bit sets the direction of the bus master transfer: when set to zero, PCI bus master reads are performed. When set to one, PCI bus master writes are performed:<br>0 = PCI bus master read<br>1 = PCI bus master write<br>While a synchronous DMA transfer is in progress, this bit will be READ ONLY. The bit will return to read/write once the synchronous DMA transfer has been completed or halted.<br>This bit must NOT be changed when the bus master function is active |
| **Bit 2-1** | **Rsv** | **Reserved** |
| **Bits 0** | **SSBM** | **Stop/Start Bus Master.** Writing a '1' to this bit enables bus master operation of the controller. Bus master operation begins when this bit is detected changing from a zero to a one. The controller will transfer data between the IDE device and memory only when this bit is set. Master operation can be halted by writing a '0' to this bit. All state information is lost when a '0' is written;. Master mode operation cannot be stopped and then resumed. If this bit is reset while bus master operation is still active (i.e., Bit 0= 1 in the Bus Master IDE Status Register for that IDE channel) and the drive has not yet finished its data transfer (bit 2=0 in the channel's Bus Master IDE Status Register), the Bus Master command is said to be aborted and data transferred from the drive may be discarded before being written to the system memory. This bit is intended to be reset after the data transfer is completed, as indicated by either the Bus Master IDE Active bit or the Interrupt bit of the IDE Status register for that IDE channel being set, or both. |

### *11.6.2. IDE STATUS REGISTER*

This 8-bit Register is addressed at offset Base + 02h for the Primary IDE Channel and Base + 0Ah for the Secondary IDE Channel.

This register provides status information about the IDE device and state of the IDE DMA transfer. Table 11-7. describes IDE Interrupt Status and Bus Master IDE Active bit states after a DMA transfer has been started.

The IDE Status Register is illustrated in the following table.

*IDE_COM*

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SO | D1DMA | D0DMA | Rsv | | RWI | RWE | RWMI |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **SO** | **Simplex only.** This is hardwired to'0'. |
| **Bit 6** | **D1DMA** | **Drive 1 DMA Capable.** This read/write bit is set by device dependent code (BIOS or device driver) to indicate that drive 1 for this channel is capable of DMA transfers, and that the controller has been initialised for optimum performance. |
| **Bit 5** | **D0DMA** | **Drive 0 DMA Capable.** This read/write bit is set by device dependent code (BIOS or device driver) to indicate that drive 0 for this channel is capable of DMA transfers, and that the controller has been initialised for optimum performance. |
| **Bits 4-3** | **Rsv** | **Reserved.** These bits return '0' when read. |
| **Bit 2** | **RWI** | **Read/Write Interrupt.** This bit is set by the rising edge of the IDE interrupt line. This bit is cleared when a '1' is written to it by software. Software can use this bit to determine if an IDE device has asserted its interrupt line. When this bit is read as a one, all data transferred from the drive is visible in system memory. For further details see Table 11-7. |
| **Bit 1** | **RWE** | **Read/Write Error.** This bit is set when the controller encounters an error in transferring data to/from memory. The exact error condition is bus specific and can be determined in a bus specific manner. This bit is cleared when a '1' is written to it by software. |
| **Bit 0** | **RWMI** | **Read/Write Bus Master IDE Active.** This bit is set to 1 when bit 0 in the Command register is set to 1. This bit is cleared (set to 0) when the last transfer for a region is performed, where EOT for that region is set in the region descriptor. It is also cleared when the Start bit is cleared in the Command register. When this bit is read as a zero, all data transferred from the drive during the previous bus master command is visible in system memory, unless the bus master command was aborted. For further details seeTable 11-7. |

**Table 11-7. Interrupt/Activity Status Combinations**

| Bit 2 | Bit 0 | Description |
|---|---|---|
| 0 | 1 | DMA transfer is in progress. No interrupt has been generated by the IDE device. |
| 1 | 0 | The IDE device generated an interrupt and the Physical Region Descriptors exhausted.This is normal completion where the size of the physical memory regions is equal to the IDE device transfer size. |
| 1 | 1 | The IDE device generated an interrupt. The controller has not reached the end of the physical memory regions. This is a valid completion case when the size of the physical memory regions is larger than the IDE device transfer size. |
| 0 | 0 | Error condition. If the IDE DMA Error bit is 1, there is a problem transferring data to/from |

### *11.7.3. DESCRIPTOR TABLE POINTER REGISTER*

This 32-bit Register is addressed at offset Base I/O+ 04h for the Primary IDE Channel and Base I/O+ 0Ch for the Secondary IDE Channel.

This register provides the base memory address of the Descriptor Table. The Descriptor Table must be DWord aligned and must not cross a 4-Kbyte boundary in memory.

*DT_Point*

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BADT | | | | | | | | | | | | | | | |
| Default value after reset = 00h | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BADT | | | | | | | | | | | | | | Rsv | |
| Default value after reset = 00h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 31-2** | **BADT** | **Base address of Descriptor table.** This field corresponds to A[31-2]. |
| **Bits 1-0** | **Rsv** | **Reserved** |

The Descriptor Table must be Dword aligned. The Descriptor Table must not cross a 64K boundary in memory.

## 11.8. OPERATION

### *11.8.1. STANDARD PROGRAMMING SEQUENCE*

To initiate a bus master transfer between memory and an Hard Disk device, the following steps are required:

1) Software prepares a PRD Table in system memory. Each PRD is 8 bytes long and consists of an address pointer to the starting address and the transfer count of the memory buffer to be transferred. In any given PRD Table, two consecutive PRDs are offset by 8-bytes and are aligned on a 4-byte boundary.

2) Software provides the starting address of the PRD Table by loading the PRD Table Pointer Register. The direction of the data transfer is specified by setting the Read/Write Control bit. Clear the Interrupt bit and Error bit in the Status register.

3) Software issues the appropriate DMA transfer command to the disk device.

4) Engage the bus master function by writing a '1' to the Start bit in the Bus Master IDE Command Register for the appropriate channel.

5) The controller transfers data to/from memory responding to DMA requests from the IDE device.

6) At the end of the transfer the IDE device signals an interrupt.

7) In response to the interrupt, software resets the Start/Stop bit in the command register. It then reads the controller status and then the drive status to determine if the transfer completed successfully.

## 11.9. DATA SYNCHRONIZATION

When reading data from an IDE device, that data may be buffered by the IDE controller before using a master operation to move the data to memory. The IDE device driver in conjunction with the IDE controller is responsible for guaranteeing that any buffered data is moved into memory before the data is used.

The IDE device driver is required to do a read of the controller Status register after receiving the IDE interrupt. If the Status register returns with the Interrupt bit set, then the driver knows that the IDE device generated the interrupt (important for shared interrupts) and that any buffered data has been flushed to memory. If the Interrupt bit is not set, then the IDE device did not generate the interrupt and the state of the data buffers is unknown.

When the IDE controller detects a rising edge on the IDE device interrupt line (INTRQ), it is required to:

- Flush all buffered data
- Set the Interrupt bit in the controller Status register
- Guarantee that a read to the controller Status register does not complete until all buffered data has been written to memory.

Another way to view this requirement is that the first read to the controller Status register in response to the IDE device interrupt must return with the Interrupt bit set and with the guarantee that all buffered data has been written to memory.

### *11.9.1. STATUS BIT INTERPRETATION*

The table below gives a description of how to interpret the Interrupt and Active bits in the Controller status register after a DMA transfer has been started.

| Interrupt bit | Active bit | Description: |
|---|---|---|
| 0 | 1 | DMA transfer is in progress. No interrupt has been generated by the IDE device. |
| 1 | 0 | The IDE device generated an interrupt. The controller exhausted the Physical Region Descriptors. This is the normal completion case where the size of the physical memory regions was equal to the IDE device transfer size. |
| 1 | 1 | The IDE device generated an interrupt. The controller has not reached the end of the physical memory regions. This is a valid completion case where the size of the physical memory regions was larger than the IDE device transfer size. |
| 0 | 0 | This bit combination signals an error condition. If the Error bit in the status register is set, then the controller has some problem transferring data to/from memory. Specifics of the error have to be determined using bus-specific information. If the Error bit is not set, then the PRD's specified a smaller size than the IDE transfer size. |

## 11.10. ERROR CONDITIONS

IDE devices are sector based mass storage devices. The drivers handle errors on a sector by sector basis; either a sector is transferred successfully or it is not. If the IDE DMA slave device never completes the transfer due to a hardware or software error, the Bus Master IDE command will eventually be stopped (by setting Command Start bit to zero) when the driver times out the disk transaction. Information in the IDE device registers will help isolate the cause of the problem.

If the controller encounters an error while doing the bus master transfers, it will stop the transfer (i.e. reset the Active bit in the Command register) and set the ERROR bit in the Status register. The controller does not generate an interrupt when this happens. The device driver can use device specific information (e.g. PCI Configuration Space Status register) to determine what caused the error.

Whenever a requested transfer does not complete properly, information in the IDE device registers (Sector Count) can be used to determine how much of the transfer was completed and to construct a new PRD table to complete the requested operation. In most cases the existing PRD table can be used to complete the operation.

### 11.11. PCI SPECIFICS

Bus master IDE controllers built to attach to a PCI bus must have the following characteristics:

1) The Class Code in PCI configuration space indicates IDE device and bit 7 of the Programming Interface register (offset 0x09) in PCI configuration space must be set to 1 to indicate that the device supports the Master IDE capability.

2) The control registers for the controller are allocated via the devices Base Address register at offset 0x20 in PCI configuration space.

3) In the controller Status register the Error bit will be set and the Active bit reset if any of the following conditions occur on the PCI bus while the controller is doing a master operation on the bus. The exact cause can be determined by examining the Configuration Space Status register.

| Error Condition | Configuration Space Status bits |
|---|---|
| Target Abort | Any time bit 12 of the Config Space Status register is set. |
| Master Abort | Any time bit 13 of the Config Space Status register is set. |
| Data Parity | Any time bit 8 of the Config Space Status register is set. |
| Error Detected | |

# 12. LOCAL BUS INTERFACE

## 12.1. INTRODUCTION

The Local Bus interface of the STPC provides a low latency bus to external peripheral cards at HCLK. The Local Bus may operate in asychronous or synchronous modes through the 22-bit address and 16-bit data bus.

The Local Bus interface supports up to two memory devices and four I/O devices. It can support up to 32 MBytes of memory for each of the memory chip selects and from 4 Byte to 1 KByte of I/O space for each of the I/O devices. All the chip select timings are individually programmable. This interface can be accessed only by the CPU.

The memory addresses are predefined for the memory chip selects. The first bank of the memory is intended to be used as a boot device.

The starting address for each IO chip select is programmable at 4 Byte boundary. The access range for each of the chip select is also programmable. The size varies from 4 Bytes to 1 KByte.

## 12.2. LOCAL BUS REGISTERS

The Local Bus configuration registers can be categorised into three groups:

1. Address Decode Registers,
2. Timing Registers,
3. Control Register.

All registers, except the CONTROL and IOWIDTHRegister, are 16-bit wide. These registers are accessible only by the CPU. All registers are accessed through I/O Port 22h and Port 23h. Port 22h is used as the index to the register bank and Port 23h is used as the data port. This way, the CPU can access only 8 bits of register data at a time, so two accesses are required to completely read or write the 16-bit registers. The lower and upper halves of the 16-bit registers have the same index values. First access after reset with the given index will map to the lower Byte of the register. The next access with the same index will access the upper Byte. For the 8-bit registers (CONTROL Register Index 1Ch and IOWIDTH Register Index 1Eh) data must be written and read twice as they are seen as 16-bit registers. Table 12-1. below shows Local Bus Register Indices.

**Table 12-1. Local Bus Register Indices**

| Name | Index | Function |
|---|---|---|
| IOAREG0 | 10h | Address Decode |
| IOAREG1 | 11h | |
| IOAREG2 | 12h | |
| IOAREG3 | 13h | |
| IOMREG0 (Slot 0 & 1) | 14h | |
| IOMREG1 (Slot 2 & 3) | 15h | |
| TIMEBANK0 | 16h | Timing |
| TIMEBANK1 | 17h | |
| TIMEIO0 | 18h | |
| TIMEIO1 | 19h | |
| TIMEIO2 | 1Ah | |
| TIMEIO3 | 1Bh | |
| CONTROL | 1Ch | Control |
| IOWIDTH | 1Eh | |

There are two other ways in which these registers can be accessed. These approaches are described in the Control Register Section 12.5.1.

## 12.3. LOCAL BUS ADDRESS DECODE REGISTERS

### *12.3.1. I/O SLOT BASE ADDRESS REGISTER 0*

This 16-bit register defines the starting address (4 Bytes or 32-bit DWORD) and I/O address spaces mapped on the Local Bus. The base address for I/O slot 0 is specified in register IOAREG0. Any accesses that hit onto the address range defined by this register and its mask (in the corresponding low Byte of the IOMREG0 register) asserts the IOCS0# and will prevent the cycle being forwarded onto the PCI bus.

*IOAREG0* Access = 0022h/0023h Regoffset = 10h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SA | | | | | | | | | | | | | | Rsv | |
| Default value after reset = 0FFh | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 15-2 | SA | Starting Address aligned to 4 I/O locations. |
| Bits 1-0 | Rsv | **Reserved.** |

Programming notes:

The two memory address spaces (for Memory Devices devices) have a fixed range:

- FF000000h - FFFFFFFFh for the Boot device

- FE000000h - FEFFFFFFh for the second device.

### *12.3.2. I/O SLOT BASE ADDRESS REGISTER 1*

This 16-bit register defines the starting address (4 Bytes or 32-bit DWORD) and I/O address spaces mapped on the Local Bus. The base address for I/O slot 1 is specified in register IOAREG1. Any accesses that hit onto the address range defined by this register and its mask (in the corresponding high Byte of the IOMREG0 register) asserts the IOCS1# and will prevent the cycle being forwarded onto the PCI bus.

*IOAREG1* Access = 0022h/0023h Regoffset = 11h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SA | | | | | | | | | | | | | | Rsv | |
| Default value after reset = 0FFh | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 15-2 | SA | Starting Address aligned to 4 I/O locations. |
| Bits 1-0 | Rsv | **Reserved.** |

### *12.3.3. I/O SLOT BASE ADDRESS REGISTER 2*

This 16-bit register defines the starting address (4 Bytes or 32-bit DWORD) and I/O address spaces mapped on the Local Bus. The base address for I/O slot 2 is specified in register IOAREG2. Any accesses that hit onto the address range defined by this register and its mask (in the corresponding low Byte of the IOMREG1 register) asserts the IOCS2# and will prevent the cycle being forwarded onto the PCI bus.

*IOAREG2* Access = 0022h/0023h Regoffset = 12h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SA | | | | | | | | | | | | | | Rsv | |
| Default value after reset = 0FFh | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 15-2 | SA | Starting Address aligned to 4 I/O locations. |
| Bits 1-0 | Rsv | **Reserved.** |

### *12.3.4. I/O SLOT BASE ADDRESS REGISTER 3*

This 16-bit register defines the starting address (4 Bytes or 32-bit DWORD) and I/O address spaces mapped on the Local Bus. The base address for I/O slot 3 is specified in register IOAREG3. Any accesses that hit onto the address range defined by this register and its mask (in the corresponding high Byte of the IOMREG1 register) asserts the IOCS3# and will prevent the cycle being forwarded onto the PCI bus.

*IOAREG3* Access = 0022h/0023h Regoffset = 13h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SA | | | | | | | | | | | | | | Rsv | |
| Default value after reset = 0FFh | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 15-2 | SA | Starting Address aligned to 4 I/O locations. |
| Bits 1-0 | Rsv | **Reserved.** |

### *12.3.5. I/O SLOT MASK REGISTER 0*

The address mask register IOMREG0 defines the size of the two first I/O slots. The 8-bit address mask for each slot will mask the address that we do not want to compare during the address decoding process. The 8-bit mask will filter bit 9:2 of the starting address specified in the corresponding IOAREG, allowing a selection of 4, 8, 16, 32, 64, 128, 256, 512 or 1K continuous or non continuous locations.

*IOMREG0* Access = 0022h/0023h Regoffset = 14h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AMIO1 | | | | | | | | AMIO0 | | | | | | | |
| Default value after reset = | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 15-8 | AMIO1 | **Address mask for I/O Slot 1.** |
| Bits 7-0 | AMIO0 | **Address mask for I/O Slot 0.** |

**Table 12-2. 8 bit Address Mask**

| 8-bit Address Mask | I/O Space Size (Bytes) |
|---|---|
| 0000 0000 | 4 |
| 0000 0001 | 8 |
| 0000 0011 | 16 |
| 0000 0111 | 32 |
| 0000 1111 | 64 |
| 0001 1111 | 128 |
| 0011 1111 | 256 |
| 0111 1111 | 512 |
| 1111 1111 | 1024 |

Note: To define an address range requires that the base address be aligned to the range (i.e. a whole multiple of the width). See Table 12-2.

Example; I/O Width = 32 (Mask = 07)

I/O Range = X .... X + 31
Then X must be an whole multiple of 32.

### *12.3.6. I/O SLOT MASK REGISTER 1*

The address mask register IOMEREG1 defines the size of I/O slots 2 & 3. The 8-bit address mask for each slot will mask the address that we do not want to compare during the address decoding process. The 8-bit mask will filter bit 9:2 of the starting address specified in the corresponding IOAREG allowing a selection of 4, 8, 16, 32, 64, 128, 256, 512 or 1K continuous or non continuous locations.

*IOMREG1* Access = 0022h/0023h Regoffset = 15h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AMIO3 | | | | | | | | AMIO2 | | | | | | | |
| Default value after reset = | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 15-8 | | Address mask for I/O Slot 3. |
| Bits 7-0 | | Address mask for I/O Slot 2. |

## 12.4. LOCAL BUS TIMING REGISTERS

### *12.4.1. MEMORY TIMING TEMPLATE REGISTER 0*

This register defines the timing template for accessing Memory Devices bank 0. The timing is programmed with reference to the host clock period as a time unit.

Timing for FLASH devices should take into account access times of 60ns and 150ns for the Boot Memory.

*TIMEBANK0* Access = 0022h/0023h Regoffset = 16h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DSH | UARS | CHT | | | CAT | | | | | | | | CST | | |
| Default value after reset = | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 15 | DSH | **Discard the Setup and Hold parameters for read operations.** |
| Bit 14 | UARS | **Use asynchronous ready signal for command termination.** The asynchronous ready enable bit sets the Local Bus logic to wait for an external ready signal before closing the current cycle. |
| Bits 13-11 | CHT | **Command Hold time,** and is determined as follows:<br>Command hold time = (5+Vh) x T<br>Where Vh = Register value for the Hold time<br>T = HCLK period. |
| Bits 10-3 | CAT | **Command Active time,** and is determined as follows:<br>Command active time = (2+Va) x T<br>Where Va = Register value for the Active time<br>T = HCLK period. |
| Bits 2-0 | CST | **Command Setup time,** and is determined as follows:<br>Command setup time = (4+Vs) x T<br>Where Vs = Register value for the Setup time<br>T = HCLK period. |

### *12.4.2. MEMORY TIMING TEMPLATE REGISTER 1*

This register defines the timing template for accessing Memory Devices bank 1. The timing is programmed with reference to the host clock period as a time unit.

Timing for FLASH devices should take into account access times of 60ns and 150ns for the Boot Memory.

*TIMEBANK1* Access = 0022h/0023h Regoffset = 17h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DSH | UARS | CHT | | | CAT | | | | | | | | CST | | |
| Default value after reset = | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 15 | DSH | **Discard the Setup and Hold parameters for read operations.** |
| Bit 14 | UARS | **Use asynchronous ready signal for command termination.** The asynchronous ready enable bit sets the Local Bus logic to wait for an external ready signal before closing the current cycle. |
| Bits 13-11 | CHT | **Command Hold time,** and is determined as follows:<br>Command hold time = (5+Vh) x T<br>Where Vh = Register value for the Hold time<br>T = HCLK period. |
| Bits 10-3 | CAT | **Command Active time,** and is determined as follows:<br>Command active time = (2+Va) x T<br>Where Va = Register value for the Active time<br>T = HCLK period. |
| Bits 2-0 | CST | **Command Setup time,** and is determined as follows:<br>Command setup time = (4+Vs) x T<br>Where Vs = Register value for the Setup time<br>T = HCLK period. |

### *12.4.3. I/O TIMING TEMPLATE REGISTER 0*

This register defines the timing template for accessing device in I/O Slot 0. The timing is programmed with reference to the host clock period as a time unit.

*TIMEIO0* Access = 0022h/0023h Regoffset = 18h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DSH | UARS | CHT | | | CAT | | | | | | | | CST | | |
| Default value after reset = | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 15 | DSH | **Discard the Setup and Hold parameters for read operations.** |
| Bit 14 | UARS | **Use asynchronous ready signal for command termination.** The asynchronous ready enable bit sets the Local Bus logic to wait for an external ready signal before closing the current cycle. |
| Bits 13-11 | CHT | **Command Hold time,** and is determined as follows:<br>Command hold time = (8+Vh) x T<br>Where Vh = Register value for the Hold time<br>T = HCLK period. |
| Bits 10-3 | CAT | **Command Active time,** and is determined as follows:<br>Command active time = (3+Va) x T<br>Where Va = Register value for the Active time<br>T = HCLK period. |
| Bits 2-0 | CST | **Command Setup time,** and is determined as follows:<br>Command setup time = (4+Vs) x T<br>Where Vs = Register value for the Setup time<br>T = HCLK period. |

### *12.4.4. I/O TIMING TEMPLATE REGISTER 1*

This register defines the timing template for accessing device in I/O Slot 1. The timing is programmed with reference to the host clock period as a time unit.

*TIMEIO1* Access = 0022h/0023h Regoffset = 19h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DSH | UARS | CHT | | | CAT | | | | | | | | CST | | |
| Default value after reset = | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 15 | DSH | **Discard the Setup and Hold parameters for read operations.** |
| Bit 14 | UARS | **Use asynchronous ready signal for command termination.** The asynchronous ready enable bit sets the Local Bus logic to wait for an external ready signal before closing the current cycle. |
| Bits 13-11 | CHT | **Command Hold time,** and is determined as follows:<br>Command hold time = (8+Vh) x T<br>Where Vh = Register value for the Hold time<br>T = HCLK period. |
| Bits 10-3 | CAT | **Command Active time,** and is determined as follows:<br>Command active time = (8+Va) x T<br>Where Va = Register value for the Active time<br>T = HCLK period. |
| Bits 2-0 | CST | **Command Setup time,** and is determined as follows:<br>Command setup time = (4+Vs) x T<br>Where Vs = Register value for the Setup time<br>T = HCLK period. |

### *12.4.5. I/O TIMING TEMPLATE REGISTER 2*

This register defines the timing template for accessing device in I/O Slot 2. The timing is programmed with reference to the host clock period as a time unit.

*TIMEIO2* Access = 0022h/0023h Regoffset = 1Ah

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DSH | UARS | CHT | | | CAT | | | | | | | | CST | | |
| Default value after reset = | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 15 | DSH | **Discard the Setup and Hold parameters for read operations.** |
| Bit 14 | UARS | **Use asynchronous ready signal for command termination.** The asynchronous ready enable bit sets the Local Bus logic to wait for an external ready signal before closing the current cycle. |
| Bits 13-11 | CHT | **Command Hold time,** and is determined as follows:<br>Command hold time = (8+Vh) x T<br>Where Vh = Register value for the Hold time<br>T = HCLK period. |
| Bits 10-3 | CAT | **Command Active time,** and is determined as follows:<br>Command active time = (3+Va) x T<br>Where Va = Register value for the Active time<br>T = HCLK period. |
| Bits 2-0 | CST | **Command Setup time,** and is determined as follows:<br>Command setup time = (4+Vs) x T<br>Where Vs = Register value for the Setup time<br>T = HCLK period. |

### *12.4.6. I/O TIMING TEMPLATE REGISTER 3*

This registers defines the timing template for accessing device in I/O Slot 3. The timing is programmed with reference to the host clock period as a time unit.

*TIMEIO3* Access = 0022h/0023h Regoffset = 1Bh

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DSH | UARS | CHT | | | CAT | | | | | | | | CST | | |
| Default value after reset = | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bit 15 | DSH | **Discard the Setup and Hold parameters for read operations.** |
| Bit 14 | UARS | **Use asynchronous ready signal for command termination.** The asynchronous ready enable bit sets the Local Bus logic to wait for an external ready signal before closing the current cycle. |
| Bits 13-11 | CHT | **Command Hold time,** and is determined as follows:<br>Command hold time = (8+Vh) x T<br>Where Vh = Register value for the Hold time<br>T = HCLK period. |
| Bits 10-3 | CAT | **Command Active time,** and is determined as follows:<br>Command active time = (3+Va) x T<br>Where Va = Register value for the Active time<br>T = HCLK period. |
| Bits 2-0 | CST | **Command Setup time,** and is determined as follows:<br>Command setup time = (4+Vs) x T<br>Where Vs = Register value for the Setup time<br>T = HCLK period. |

## 12.5. LOCAL BUS CONTROL REGISTER

### *12.5.1. CONTROL REGISTER*

*CONTROL* Access = 0022h/0023h Regoffset = 1Ch

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RAM | | 32AFME | CEB1 | CEB0 | WEB1 | WEB0 | RMBAE |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 7-6 | RAM | **Register Access Map.** See Table 12-3. |
| Bit 5 | 32AFME | **32 bit access to Memory Devices Enable.** Setting this bit enables 32-bit access to both bank1 and bank0. |
| Bit 4 | CEB1 | **Cache Enable for Bank1.** |
| Bit 3 | CEB0 | **Cache Enable for Bank0.** |
| Bit 2 | WEB1 | **Write Enable for Bank1.** |
| Bit 1 | WEB0 | **Write Enable for Bank0.** |
| Bit 0 | RMBAE | **Real Mode Boot Access Enable.** When set this bit enables boot access in Real Mode by mapping 000C0000h to FFFC0000h and 000FFFFFh to FFFFFFFFh. |

**Table 12-3. Register Access Map**

| Bit 7 | Bit 6 | Register Access |
|---|---|---|
| 0 | 0 | Access with same index will alternately map to lower and upper and lower Bytes of the 16 bit register as described in Section 12.2. |
| 0 | 1 | Register accesses will always map to the lower Byte. |
| 1 | 0 | Register accesses will always map to the upper Byte. |
| 1 | 1 | Access with same index will alternately map to lower and upper and lower Bytes of the 16 bit register as described in Section 12.2. |

### *12.5.2. IO WIDTH REGISTER*

This is an 8-bit register where the four less significant bits are used to tell the local bus if an 8-bit or 16-bit peripheral is attached to one of the four I/O slots.

*IOWIDTH* Access = 0022h/0023h Regoffset = 1Eh

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | | | IOW3 | IOW2 | IOW1 | IOW0 |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 7-4 | Rsv | **Reserved.** |
| Bit 3 | IOW3 | If set to 1, the I/O 3 is a 16 bit peripheral, if set to 0 it is an 8 bit peripheral. |
| Bit 2 | IOW2 | If set to 1, the I/O 2 is a 16 bit peripheral, if set to 0 it is an 8 bit peripheral. |
| Bit 1 | IOW1 | If set to 1, the I/O 1 is a 16 bit peripheral, if set to 0 it is an 8 bit peripheral. |
| Bit 0 | IOW0 | If set to 1, the I/O 0 is a 16 bit peripheral, if set to 0 it is an 8 bit peripheral. |

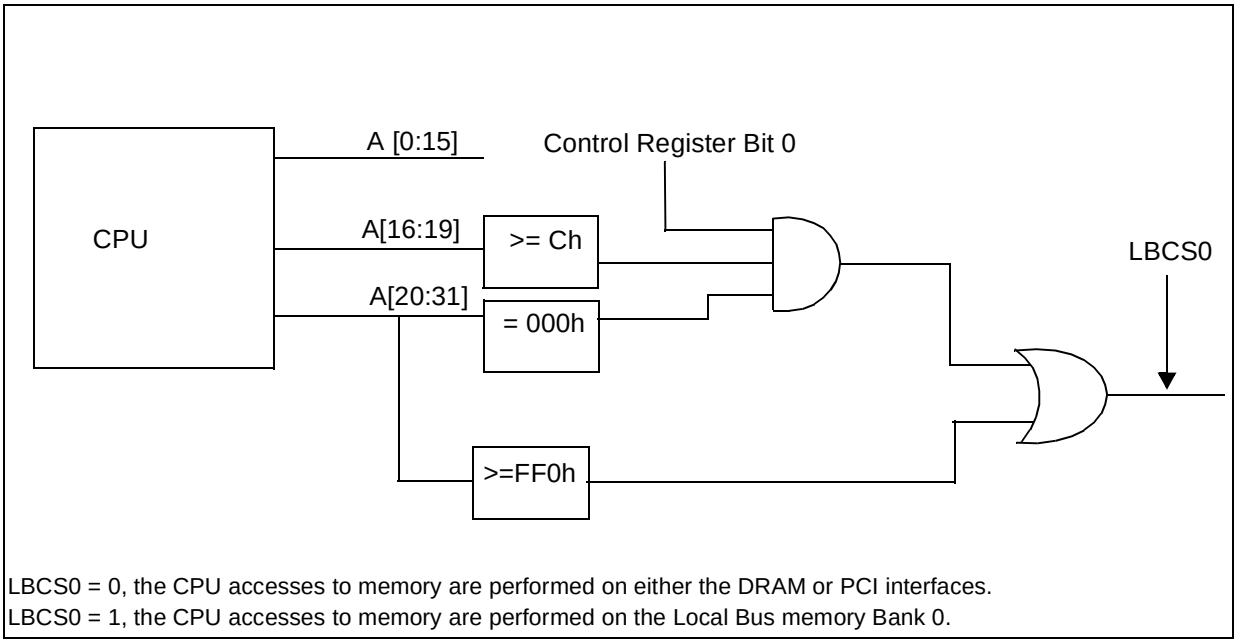
### 12.6. CHIP SELECT MEMORY MAP

The address mapping supported by the Local Bus interface is summarised in Table 12-4. Memory address ranges are mapped at fixed addresses while the I/O devices can be mapped from 1 DWord (double word) to 256 DWord inside a 16-MByte segment.

**Table 12-4. Local Bus Address Mapping**

| **Memory Devices Chip Select** | **Bank 0** | | **Bank 1** | |
|---|---|---|---|---|
| Address Range | FF000000h-FFFFFFFFh | | FE000000h-FEFFFFFFh | |
| Address Space | 16 MBytes | | 16 MBytes | |
| Boot Address Space (Real Mode) | 000C0000h-000FFFFFH | | - | |
| | | | | |
| I/O Control Chip Select | IOCS#0 | IOCS#1 | IOCS#2 | IOCS#3 |
| Address Range | 0000h-FFFFh | 0000h-FFFFh | 0000h-FFFFh | 0000h-FFFFh |
| Address Space | 4 Bytes - 1KByte | 4 Bytes - 1KByte | 4 Bytes - 1KByte | 4 Bytes - 1KByte |

Memory access to the Local Bus is based on the following logic scheme, see Figure 12-1.




LBCS0 = 0, the CPU accesses to memory are performed on either the DRAM or PCI interfaces.
LBCS0 = 1, the CPU accesses to memory are performed on the Local Bus memory Bank 0.

**Figure 12-1. Memory Bank 0 Access Logic**

# 13. GPIO INTERFACE

## 13.1. INTRODUCTION

The GPIO Interface provides a general purpose 16-bit I/O facility, using 16 dedicated device pins. It is organised using two blocks of 8-bit Registers (tabulated below), one block located at Base address 0320h for I/O Ports 0 to 7, the other at Base address 0328h for I/O Ports 8 to 15..

**Table 13-1. GPIO Port Registers**

| Register Name | Mnemonic | Offset Value | Read/Write | Default Value (h) |
|---|---|---|---|---|
| Port Direction Control Register (Base+00h): | portDirCtrl | Base + 000 | read/write | FF |
| Read Port Control | readPortCtrl | Base + 001 | read/write | 00 |
| Read Register | readReg | Base + 010 | read/write | 00 |
| Interrupt Unmask | intrUnMask | Base + 011 | read/write | 00 |
| Interrupt Edge Select | intrEdgeSelect | Base + 100 | read/write | 00 |
| Clear Interrupt | clearIntr | Base + 101 | write only | - |
| GPIO Port | GPIOport | Base + 110 | read/write | - |
| Strap Register | strapReg | Base + 111 | read only | Read at Reset from Strap |

Each block is an 8-bit slave device which can be integrated onto an ISA-type bus. Each block has the following features:

- Built-in debounce logic for each Input port.
- Each GPIO port is configurable for Rise or Fall edge interrupt generation.
- Strap recording on all 16 ports, 8 ports per block.

Each GPIO port can be configured as an input or an output simply by programming the associated port direction control register. All GPIO ports are configured as inputs at reset, which also latches the input levels into the Strap Registers. The input states of the ports are thus recorded automatically at reset, and this can be used as a strap register anywhere in the system.

All the GPIO ports can be configured to generate rise or fall edge interrupts, but this can only be done when the ports are configured as inputs. These interrupt states can be cleared from the Interrupt Edge Select Register by writing to the Clear Interrupt Register (Base+05h). In cascade mode it is also necessary to clear the interrupt of the Master GPIO.

## 13.2. GPIO BASE ADDRESS

The following table details the programming for the PCI config space to access GPIO. Note that the GPIOs are disabled at reset

**Table 13-2. PCI South Bridge Config Space for GPIO Programming**

| Offset | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 40 | Device Control | Reserved | | | | See GPIO_MDC[1] in Section 9.8.17. | See GPIO_SDC[2] in Section 9.8.17. | See GPIO[3] in Section 9.8.17. | See Section 9.7.6. |
| 44 | Device Control | GPIO Master Base Address [7:3] | | | | | 0 | 0 | 1 |
| 45 | Device Control | GPIO Master Base Address [15:8] | | | | | | | |
| 46 | Device Control | GPIO Slave Base Address [7:3] | | | | | 0 | 0 | 1 |
| 47 | Device Control | GPIO Slave Base Address [15:8] | | | | | | | |
| 48 | Device Control | GPIO Debounce Count [7:0] | | | | | | | |
| 49 | Device Control | GPIO Debounce Count [15:8] | | | | | | | |
| 50 | Device Control | GPIO Debounce Count [23:16] | | | | | | | |
| 51 | Device Control | GPIO Debounce Count [31:24] | | | | | | | |

Note 1. This is a separate control for Master input port debounce control enable/disable.
Note 2. This is a separate control for Slave input port debounce control enable/disable.
Note 3. Enabling GPIO will enable both master and slave GPIO controllers.

### *13.2.1.* GPIO MASTER BASE ADDRESS

This 16-bit register sets the base addrress of the eight Master GPIO ports.

GPIO_MBA Access = Section Table 13-2. Regoffset = 044h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| GPIO_MBA | | | | | | | |
| Default value after reset = 0321h | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| GPIO_MBA | | | | | 0 | 0 | 1 |
| Default value after reset = 0321h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 15-3 | GPIO_MBA | GPIO Master Base Address [157:3]. See Section 13.1. |
| Bits 2-0 | Rsv | These bits are hardwired to 001 |

### *13.2.2.* GPIO SLAVE BASE ADDRESS

This 16-bit register sets the base addrress of the eight Master GPIO ports.

GPIO_SBA Access = Section Table 13-2. Regoffset = 046h

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| GPIO_SBA | | | | | | | |
| Default value after reset = 0329h | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| GPIO_SBA | | | | | 0 | 0 | 1 |
| Default value after reset = 0329h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 15-3 | GPIO_SBA | GPIO Master Base Address [15:3]. See Section 13.1. |
| Bits 2-0 | Rsv | These bits are hardwired to 001 |

### *13.2.3. GPIO DEBOUNCE COUNT REGISTER*

This 32-bit register sets the debounce counter. The debounce counter is disabled at reset.

GPIO_DC Access = Section Table 13-2. Regoffset = 048h

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO_DC | | | | | | | | | | | | | | | |
| Default value after reset = 00000000h | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO_DC | | | | | | | | | | | | | | | |
| Default value after reset = 00000000h | | | | | | | | | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 31-0 | GPIO_DC | **GPIO Debounce Count:** |

## 13.3. REGISTER DESCRIPTION

The GPIO requires eight address lines to access all functions. The registers are explained in the following sections.

### *13.3.1. PORT DIRECTION CONTROL REGISTER (BASE+00H):*

This 8-bit register sets the direction, input or output, of each of the eight GPIO ports. After reset this register defaults to 0xFF, setting all ports to the input mode.

portDirCtrl Access = Section 13.1. Regoffset = 000h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Port 7 (15) | Port 6 (14) | Port 5 (13) | Port 4 (12) | Port 3 (11) | Port 2 (10) | Port 1 (9) | Port 0 (8) |
| Default value after reset = FFh | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 7-0 | | 0 = Output, 1= Input. |

### *13.3.2. READ PORT CONTROL REGISTER (BASE+01H):*

This is an 8-bit register which decides whether the data read at address Base+06H, for the ports which are configured as outputs, is the data at the GPIO Port or is the data from the read register at address Base+02h.

For all ports which are configured as Inputs, the read data at address Base+06h always returns the data at the GPIO Port.

For any port which is configured as an Output, the following holds true.

A ONE in any bit position returns the read data, at address Base+06h, from the Read Register, which is accessible at address Base+02h.

A ZERO in any bit position returns the read data, at address Base+06h, from the GPIO Port.

readPortCtrl Access = Section 13.1. Regoffset = 001h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Port 7 (15) | Port 6 (14) | Port 5 (13) | Port 4 (12) | Port 3 (11) | Port 2 (10) | Port 1 (9) | Port 0 (8) |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 7-0 | | 1: Returns the read data, at address Base+06h, from the Read Register, which is accessible at address Base+02h.<br>0: Returns the read data, at address Base+06h, from the GPIO Port. |

### *13.3.3. READ REGISTER (BASE+02H):*

This is an 8-bit register which is used to return the read data for the ports which are configured as Outputs, provided also that their corresponding bits in the Read Port Control Register are set to 1.

readReg Access = Section 13.1. Regoffset = 010h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Port 7 (15) | Port 6 (14) | Port 5 (13) | Port 4 (12) | Port 3 (11) | Port 2 (10) | Port 1 (9) | Port 0 (8) |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 7-0 | | Read data register for Ports configured as Outputs. |

### *13.3.4. INTERRUPT UNMASK REGISTER (BASE+03H):*

This is an 8-bit register with each bit controlling the interrupt mask for the corresponding port. After reset this register is cleared, disabling all interrupt generation. Interrupts can be un-masked only for the ports which are configured as inputs. Clearing the mask for the ports which are configured as outputs has no effect.

intrUnMask Access = Section 13.1. Regoffset = 011h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Port 7 (15) | Port 6 (14) | Port 5 (13) | Port 4 (12) | Port 3 (11) | Port 2 (10) | Port 1 (9) | Port 0 (8) |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 7-0 | | Write interrupt mask bits for selected input ports. |

### *13.3.5. INTERRUPT EDGE REGISTER (BASE+04H):*

This is an 8-bit register. Each bit controls the trigger for interrupt generation for the corresponding ports. This has no effect if its corresponding interrupt is masked. After reset this register is cleared, which means that if the port's interrupt is unmasked, then the interrupt trigger condition is the rising edge on the input port. Again, this register is don't care for all ports which are configured as outputs.

.

intrEdgeSelect Access = Section 13.1. Regoffset = 100h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Port 7 (15) | Port 6 (14) | Port 5 (13) | Port 4 (12) | Port 3 (11) | Port 2 (10) | Port 1 (9) | Port 0 (8) |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 7-0 | | Write interrupt edge select bits for selected input ports<br>0: rising edge<br>1: falling edge |

### *13.3.6. INTERRUPT CLEAR COMMAND (BASE+05H):*

This is a write only address location. A read will return 0FFh.

.

clearIntr Access = Section 13.1. Regoffset = 101h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Port 7 (15) | Port 6 (14) | Port 5 (13) | Port 4 (12) | Port 3 (11) | Port 2 (10) | Port 1 (9) | Port 0 (8) |
| Default value after reset = h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 7-0 | | Write 1 to clear interrupt for selected port. |

### *13.3.7. GPIO PORT REGISTER (BASE+06H):*

This is an 8-bit register which controls/reads the GPIO port. A write to this with any data will change the value on the ports which are configured as outputs. The ports configured as inputs are not affected. Whereas a read to this address will return the value at the port for those ports which are configured as inputs. For those ports which are configured as outputs, a read will return either the output value OR will return the data from the read register, depending on the programming of the readPortCtrl register.

.

GPIOport Access = Section 13.1. Regoffset = 110h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Port 7 (15) | Port 6 (14) | Port 5 (13) | Port 4 (12) | Port 3 (11) | Port 2 (10) | Port 1 (9) | Port 0 (8) |
| Default value after reset = h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 7-0 | | Write output port data, read input port data |

### *13.3.8. STRAP REGISTER (BASE+07H):*

This is an 8-bit register and is used to latch the value on the GPIO port at reset. This therefore becomes a strap register and can be used anywhere in the system. This register is read only; a write to this register has no effect.

.

strapReg Access = Section 13.1. Regoffset =111h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Port 7 (15) | Port 6 (14) | Port 5 (13) | Port 4 (12) | Port 3 (11) | Port 2 (10) | Port 1 (9) | Port 0 (8) |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| Bits 7-0 | | Read and latch GPIO port state at reset |

# 14. POWER MANAGEMENT

## 14.1. INTRODUCTION

For full information on the action of the System Management Mode (SMM), please refer to the STMicroelectronics manual for the ST486 CPU Core. This chapter describes the SMM control registers for the STPC.

The STPC provides the following hardware structures to assist the software in managing system power consumption:

– System Activity detection,
– Three power-down timers,
– Doze timer for detecting short-duration lack of system activity,
– Standby timer for detecting medium-duration lack of system activity,
– Suspend timer for detecting long-term lack of system activity,
– House-keeping activity detection,
– House-keeping timer to cope with short bursts of house-keeping activity while dozing or in standby state,
– Peripheral Activity detection,
– Peripheral timer for detecting lack of peripheral activity,
– STPCLK# modulation to adjust system performance in various system power down states, including full power-on state.

Lack of system activity for progressively longer periods of times is detected by the three power-down timers. These timers can generate a System Management Interrupt (SMI) to the CPU so that the SMM software can put the system in decreasing states of power consumption. System activity in a power-down state can generate an SMI to allow the software to bring the system back up to the full power-on state. The chipset supports up to three power-down states: Doze state, Standby state and Suspend state. These correspond to increasing levels of power savings.

The chipset can detect the presence or absence of the following System activities:

– DMA Request (DRQ),
– Interrupt Request (INTR),
– Parallel IO (PIO),
– Serial IO (SIO) ,
– Keyboard (KBD),
– Floppy Disk Controller (FDC),
– Hard Disk Controller (HDC) ,
– PCI master device,
– Programmable address range.

Each of these can be individually enabled. The presence of an enabled system activity resets the power-down timers. The chipset generates the SMI when no system activity is detected for the delay period programmed in the power-down timers. The software can then put the appropriate sub-systems in the power-down mode, request STPCLK# assertion and stop CPU and other system clocks, program the current power-down state in the chipset and set up the next timer.

The presence of an enabled system activity, when the STPC is in a power-down state, will first enable any stopped clocks, wait for a a programmable delay to allow any internal Phase Locked Loop (PLL) to stabilise and then deassert STPCLK# to enable CPU execution. The device can optionally generate an SMI to allow the SMM to bring the system back to the power-on state.

The current revision of the STPC does not implement support for stopping CPU and other system clocks.

In Doze or Standby state, a house-keeping activity, can bring the system back to full speed for a short period of time before returning back to Doze or Standby state. The chipset can detect the following house-keeping activities:

– DMA Request (DRQ),
– Interrupt Request (INTR),
– Keyboard (KBD),
– PCI master device.

The house-keeping timer determines the length of time the system will be on before returning to the original power-down state. An activity can be either a system activity or a house-keeping activity, but not both at the same time. Further, the Suspend state cannot make use of this feature.

The absence of the following peripheral activities can be enabled to cause an SMI and thus allow the software to put the unused peripherals in the power-down state, while the remainder of the system is still in full power-on state:

– Parallel IO (PIO),
– Serial IO (SIO) ,
– Keyboard (KBD),
– Floppy Disk Controller (FDC),
– Hard Disk Controller (HDC),
– A programmable address range.

Each of these can be individually enabled for inactivity detection. The presence of a peripheral activity does not reset the peripheral timer. It always times out after the programmed delay period. An SMI is generated if any enabled peripheral was not active for this time period. The device provides IO access trapping to detect access to a powered-down peripheral, so that the software can bring the peripheral to the power-on state before the access is completed.

The STPC can also carry out software transparent power management, if so enabled. In this mode of operation, doze and standby time-outs will change the CPU clock without generating an SMI. The state transitions from fully-on to doze or standby and back to fully-on will take place automatically. Also note that the suspend state can never be entered automatically but always requires software assist.

The STPC decodes the various activities listed below in Table 14-1.

**Table 14-1. Activity Detected**

| Activity | Detected via |
|---|---|
| ISA DMA masters | Low to high transition of hold request of 206 |
| PCI masters | High to low transition of any of PCIRQ2-0# |
| Parallel port | IO read/write at 378h-37Fh,<br>278h-27Fh and 3BCh-3BFh |
| Serial port | IO read/write at 3F8h-3FFh, 2F8h-2FFh, 3E8h-3EFh and 2E8h-2EFh |
| Keyboard | IO read/write at 60h, 62h, 64h and 66h |
| Floppy disk | IO read/write at 3F2h, 3F4h,<br>3F5h and 3F7h |
| Hard disk | IO read/write in 170h-177h, 376h,1F0h-1F7h and 3F6h address range as well as any bus master activity by the internal IDE controller. |

## 14.2. POWER MANAGEMENT CONTROLLER REGISTERS

### *14.2.1. TIMER REGISTER 0*

This register controls the timer for the selection of the length of timeout for the doze, standby and suspend modes.

*Timer0* Access = 0022h/0023h Regoffset = 060h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SUTT | | | STT | | | Rsv | |
| Default value after reset = undefined | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-5** | **SUTT** | **Suspend Timeout Timer**, when set to any value other than the disable value (000), this timer will generate an SMI on time out.<br>Once enabled, this timer counts down from the programmed valued. If any of the enabled system activities are detected before time out, the timer will reset and start again. These bits are encoded as given in Table 14-2.<br>The suspend timer will count whenever it is not disabled and the suspend time-out bit in the SMI status register 0 is not set to a 1. |
| **Bits 4-2** | **STT** | **Standby Timeout Timer,** when set to any value other than the disable value (000) this timer, on expiration, can either generate the SMI to the CPU or if programmed for auto-power saving (software transparent power management) mode, change the power-down state to Standby state (refer to auto-power saving mode for details of the power saving features that are enabled in the standby state). Similar to the Suspend timer, presence of an enabled system activity will reset the timer to restart counting. These bits are encoded as given in Table 14-3.<br>The standby timer will count whenever it is not disabled and the standby time-out bit in the SMI status register 0 is not set to a 1. |
| **Bits 1-0** | **Rsv** | **Reserved.** |

**Table 14-2. Suspend Timer Reset**

| Bit 7 | Bit 6 | Bit 5 | Suspend Timer reset |
|---|---|---|---|
| 0 | 0 | 0 | Disabled |
| 0 | 0 | 1 | 4 minutes |
| 0 | 1 | 0 | 8 minutes |
| 0 | 1 | 1 | 12 minutes |
| 1 | 0 | 0 | 16 minutes |
| 1 | 0 | 1 | 32 minutes |
| 1 | 1 | 0 | 48 minutes |
| 1 | 1 | 1 | 64 minutes |

**Table 14-3. Standby Timer Reset**

| Bit 4 | Bit 3 | Bit 2 | Standby Timer reset |
|---|---|---|---|
| 0 | 0 | 0 | Disabled |
| 0 | 0 | 1 | Reserved |
| 0 | 1 | 0 | 2 minutes |
| 0 | 1 | 1 | 4 minutes |
| 1 | 0 | 0 | 6 minutes |
| 1 | 0 | 1 | 8 minutes |
| 1 | 1 | 0 | 12 minutes |
| 1 | 1 | 1 | 16 minutes |

### *14.2.2. TIMER REGISTER 1*

*Timer1* Access = 0022h/0023h Regoffset = 061h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | HKT | | | PTT | | | Rsv |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **Rsv** | **Reserved.** |
| **Bits 6-4** | **HKT** | **House-keeping Timer.** This timer determines how long the PMU will be in Doze house-keeping state when an enabled house-keeping activity is detected while in doze or standby power-down states. It is encoded as given in Table 14-4.<br>The house-keeping counts only when the PMU is in one of the house-keeping states. Another house-keeping activity while the controller is in house_keeping state will reset the house-keeping timer to restart counting.<br>A system activity detection in the house_keeping state will have the same effect as if the controller was in Doze or Standby state. Either an SMI will be generated to allow the software to bring the system to power-on state or the controller will automatically transition to power-on state. The house-keeping timer and function can be disabled by masking out all activity detection via the House-keeping Enable registers. |
| **Bits 3-1** | **PTT** | **Peripheral Timeout Timer.** When set to a value other than (000) this timer on expiration, will generate an SMI if any of the enabled peripherals remained inactive during the entire period. Unlike the power-down timers, the peripheral timer does not reset due to an enabled peripheral activity. It always times out after the programmed delay. An SMI is generated only if any of the enabled peripherals were inactive during this period. This field is encoded as given in Table 14-5.<br>The peripheral timer counts whenever it is enabled. |
| **Bit 0** | **Rsv** | **Reserved.** |

**Table 14-4. House-keeping Timer Reset**

| Bit 6 | Bit 5 | Bit 4 | House-keeping Timer reset |
|---|---|---|---|
| 0 | 0 | 0 | Disabled |
| 0 | 0 | 1 | 64 micro-seconds |
| 0 | 1 | 0 | 128 micro-seconds |
| 0 | 1 | 1 | 256 micro-seconds |
| 1 | 0 | 0 | Reserved |
| 1 | 0 | 1 | 4 milli-seconds |
| 1 | 1 | 0 | 16 milli-seconds |
| 1 | 1 | 1 | 32 milli-seconds |

**Table 14-5. Peripheral Timer Reset**

| Bit 3 | Bit 2 | Bit 1 | Peripheral Timer reset |
|---|---|---|---|
| 0 | 0 | 0 | Disabled |
| 0 | 0 | 1 | 8 seconds |
| 0 | 1 | 0 | 16 seconds |
| 0 | 1 | 1 | 32 seconds |
| 1 | 0 | 0 | 64 seconds |
| 1 | 0 | 1 | 128 seconds |
| 1 | 1 | 0 | 256 seconds |
| 1 | 1 | 1 | 512 seconds |

### *14.2.3. TIMER REGISTER 2*

*Timer 2* Access = 0022h/0023h Regoffset = 08Dh

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DTT | | | Rsv | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7- 5** | **DTT** | **Doze Timeout Timer.** When set to any value other than the disable value (00), this timer, on expiration, can either generate the SMI to the CPU or if programmed for auto-power saving (software transparent power management) mode, change the power-down state to Doze state (refer to auto-power saving mode for details of the power saving features that are enabled with Doze state). Similar to the suspend timer, presence of an enabled system activity will reset the timer to restart counting. This 3-bit field is encoded as given in Table 14-6.<br>The doze timer will count whenever it is not disabled and the doze time-out bit in the SMI status register 0 is not set to a '1'. |
| **Bits 4-2** | **Rsv** | **Reserved.** |

**Table 14-6. Doze Timer Reset**

| Bit 7 | Bit 6 | Bit 5 | Doze Timer reset |
|---|---|---|---|
| 0 | 0 | 0 | Disabled |
| 0 | 0 | 1 | 50 milli-seconds |
| 0 | 1 | 0 | 100 milli-seconds |
| 0 | 1 | 1 | 500 milli-seconds |
| 1 | 0 | 0 | Reserved |
| 1 | 0 | 1 | 4 seconds |
| 1 | 1 | 0 | 8 seconds |
| 1 | 1 | 1 | 16 seconds |

### *14.2.4. SYSTEM ACTIVITY ENABLE REGISTER 0*

This is the first of the three registers that control which system activity to detect.

*Sys_Activ_en0* Access = 0022h/0023h Regoffset = 062h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DRQ | PCIM | PIO | SIO | KBD | FDC | HDC | Rsv |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **DRQ** | **DMA Request (DRQ).** |
| **Bit 6** | **PCIM** | **PCI master device (PCIM).** |
| **Bit 5** | **PIO** | **Parallel IO (PIO).** |
| **Bit 4** | **SIO** | **Serial IO (SIO).** |
| **Bit 3** | **KBD** | **Keyboard (KBD).** |
| **Bit 2** | **FDC** | **Floppy Disk Controller (FDC).** |
| **Bit 1** | **HDC** | **Hard Disk Controller (HDC).** |
| **Bit 0** | **Rsv** | **Reserved.** |

Programming notes:

When detected, the power-down timers will reload with their initial time values, or if enabled via the SMI control register, an SMI will be generated, or if programmed for auto-power down mode and in Doze or Standby power-down states, transition to power-on state will take place. Set the following bits to ‘1’ to detect the associated activity, and to ‘0’ to ignore the associated activity.

### *14.2.5. SYSTEM ACTIVITY ENABLE REGISTER 1*

This is the second of the three registers that control which system activity to detect.

*Sys_Activ_en1* Access = 0022h/0023h Regoffset = 063h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | AR0 | Rsv | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-6** | **Rsv** | **Reserved.** Must be programmed to ‘0’. |
| **Bit 5** | **AR0** | **Address range 0.** |
| **Bits 4-0** | **Rsv** | **Reserved.** Must be programmed to ‘0’. |

### *14.2.6. SYSTEM ACTIVITY ENABLE REGISTER 2*

This is the third of the three registers that control which system activity to detect.

*Sys_Activ_en2* Access = 0022h/0023h Regoffset = 064h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IRQ15-1 | IRQ0 | NMI | Rsv | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **IRQ15-1** | **IRQ15-1** detection enabled. |
| **Bit 6** | **IRQ0** | **IRQ0** detection enabled. |
| **Bit 5** | **NMI** | **NMI** detection enable. |
| **Bits 4-0** | **Rsv** | **Reserved.** |

### *14.2.7. HOUSE-KEEPING ACTIVITY ENABLE REGISTER 0*

This register controls which house-keeping activity to detect. House-keeping activities are detected only in Doze and Standby states. If enabled, a house-keeping activity reverts the system back to power-on state for a short period of time, programmed in the house-keeping timer. Set the following bits to a '1' to enable activity detection and a '0' to ignore the associated activity.

*HK_Activ_en0* Access = 0022h/0023h Regoffset = 065h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DRQ | PCI MD | KBD | IRQ15-1 | IRQ0 | NMI | Rsv | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **DRQ** | **DMA Request (DRQ)** activity |
| **Bit 6** | **PCI MD** | **PCI master** device activity |
| **Bit 5** | **KBD** | **Keyboards** (KBD) activity |
| **Bit 4** | **IRQ15-1** | **IRQ15-1** activity |
| **Bit 3** | **IRQ0** | **IRQ0** activity |
| **Bit 2** | **NMI** | **NMI** activity |
| **Bits 1-0** | **Rsv** | **Reserved.** |

### *14.2.8. HOUSE-KEEPING ACTIVITY ENABLE REGISTER 1*

This is the second house-keeping activity detection enable register.

*HK_Activ_en1* Access = 0022h/0023h Regoffset = 066h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | AR0 | Rsv | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-6** | **Rsv** | **Reserved.** Must be programmed to ‘0’. |
| **Bit 5** | **AR0** | **Address range 0**. |
| **Bits 4-0** | **Rsv** | **Reserved.** Must be programmed to ‘0’. |

### *14.2.9. PERIPHERAL INACTIVITY DETECTION REGISTER 0*

This register controls which peripheral inactivity is enabled for generating an SMI on a peripheral time-out.

*Perif_Inact0* Access = 0022h/0023h Regoffset = 067h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PIO | SIO | KBD | FDC | HDC | ARO | Rsv | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **PIO** | **Parallel IO (PIO)** activity. |
| **Bit 6** | **SIO** | **Serial IO (SIO)** activity. |
| **Bit 5** | **KBD** | **Keyboard (KBD)** activity. |
| **Bit 4** | **FDC** | **Floppy Disk Controller (FDC)** activity. |
| **Bit 3** | **HDC** | **Hard Disk Controller (HDC)** activity. |
| **Bit 2** | **AR0** | **Address range 0.** |
| **Bits 1-0** | **Rsv** | **Reserved.** Must be programmed to ‘0’. |

Programming notes:

Lack of peripheral activity for an enabled peripheral for one peripheral time-out period generates an SMI. A ‘1’ in a bit position enables the SMI generation for the associated peripheral and a ‘0’ disables it. Software can use the Peripheral Inactivity status register to determine which peripheral should be powered down.

### *14.2.10. PERIPHERAL ACTIVITY DETECTION REGISTER 0*

This register controls which peripheral accesses will cause an SMI.

*Perif_Act0* Access = 0022h/0023h Regoffset = 069h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PIO | SIO | KBD | FDC | HDC | ARO | Rsv | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **PIO** | **Parallel port (PIO)** access |
| **Bit 6** | **SIO** | **Serial port (SIO)** access |
| **Bit 5** | **KBD** | **Keyboard (KBD)** access |
| **Bit 4** | **FDC** | **Floppy Disk Controller (FDC)** access |
| **Bit 3** | **HDC** | **Hard Disk Controller (HDC)** access |
| **Bit 2** | **AR0** | **Address range 0** |
| **Bits 1-0** | **Rsv** | **Reserved.** Must be programmed to ‘0’ |

Programming notes:

Typically the power management software will detect non-usage of a peripheral device via Peripheral in-activity status registers, bring the peripheral into power down state and then enable trapping access to that peripheral via this register.

Thus when an application attempts to make use of a powered down peripheral, the access is trapped and an SMI is generated to allow software to re-power the peripheral device before allowing the access to complete. This register is the first of the two such registers.

A ‘1’ in a bit position enables SMI generation for the associate peripheral and a ‘0’ disables.

### *14.2.11. PERIPHERAL ACTIVITY DETECTION REGISTER 1*

This is the second register that controls which peripheral accesses will cause an SMI. This register is similar in functionality to Peripheral Activity detection register 0.

*Perif_Act1* Access = 0022h/0023h Regoffset = 06Ah

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-6** | **Rsv** | **Reserved.** Must be programmed to ‘0’. |

### *14.2.12. ADDRESS RANGE 0 REGISTER 0*

This register contains bits which are compared with PCI address bits 31-24 if range compare is enabled for memory cycle or compared against bits 15-8 if range compare is enabled for IO cycles.

*Add_Rang0-0* Access = 0022h/0023h Regoffset = 06Bh

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| Default value after reset = 00h | | | | | | | |

*14.2.13. ADDRESS RANGE 0 REGISTER 1*

*Add_Rang0-1* Access = 0022h/0023h Regoffset = 06Ch

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-3** | | These bits are compared with PCI address bits 23-19 if range compare is enabled for memory cycle, or compared against bits 7-3 if range compare is enabled for IO cycles. |
| **Bit 2** | | This bit is compared with PCI address bit 18 if range compare is enabled for memory cycle, or compared with address bit 2 if range compare is enabled for IO cycles and range is 4-Bytes. Otherwise this bit when 1 specifies that the range of IO address to be compared is 16-Bytes and when 0, the range is 8-Bytes. |
| **Bit 1** | | This bit is compared with PCI address bit 17 if range compare is enabled for memory cycle. Otherwise if range compare is enabled for IO cycles, this bit if 1 specifies that the range of IO address to be compared is 8/16-Bytes and when 0 the range is 4-Bytes. |
| **Bit 0** | | This bit when ‘1’ specifies that range compare should be done for memory cycles and when ‘0’, for IO cycles. |

### *14.2.14. SMI CONTROL REGISTER 0*

This register controls the generation of an SMI, as follows:

*SMI_Cont0* Access = 0022h/0023h Regoffset = 071h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | Rsv |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | | If '1' then generate SMI on Doze time-out. Otherwise if set to a '0', the hardware will transition to Doze state automatically on Doze time-out. |
| **Bit 6** | | If '1' then generate SMI on Standby time-out. Otherwise if set to a '0', the hardware will transition to Standby state automatically on Standby time-out. |
| **Bit 5** | | If '1' then generate SMI on Suspend time-out. Otherwise if set to a '0', SMI is not generated. The hardware never transitions into Suspend state by itself. |
| **Bit 4** | | If '1' then generate SMI on House-keeping time-out. Otherwise if set to a '0', the hardware will automatically transition back to the doze or standby state (which ever state it was in before entering house-keeping state). |
| **Bit 3** | | If '1' then generate SMI on detecting a house-keeping activity. Otherwise if set to a '0', and if in Doze or Standby state, the hardware will automatically transition to the associated house_keeping states for the duration programmed in the house-keeping timer. |
| **Bit 2** | | If '1' then generate SMI on detecting a system activity. Otherwise if set to a '0', and if in Doze or Standby state, the hardware will automatically transition to Power-on state on detecting a unmasked system activity. This bit will typically be set to a 1' by software on entering a power-down state so that a system activity can wake up the system. |
| **Bit 1** | | This is a write only bit. Setting this bit to a '1' sets bit-7 of the SMI status register 1 and generates an SMI. This bit however will always read back as '0'. |
| **Bit 0** | **Rsv** | **Reserved.** |

### *14.2.15. SMI STATUS REGISTER 0*

This register contains the status information pertaining to the SMI.

*SMI_Stat0* Access = 0022h/0023h Regoffset = 073h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DTO | STO | STO | HKT | HKA | SAD | PID | PAD |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **DTO** | **Doze time-out.** This bit is set to a '1' when Doze time-out occurs. An SMI will be generated if associated SMI enable bit in SMI Control register is set to a '1'. The software must write this bit to a '1' to deassert SMI#. If SMI generation has been disabled then the controller will automatically transition to Doze state. This bit will then be cleared on transition from Doze or Standby to Power-on state. |
| **Bit 6** | **STO** | **Standby time-out.** This bit will be set to a '1' when Standby time-out occurs. An SMI will be generated if the associated SMI enable bit in SMI Control register is set to a '1'. The software must write this bit to a '1' to deassert SMI#. If SMI generation has been disabled then the hardware will automatically transition to Standby state. This bit will then be cleared on transition Standby to Power-on state. |
| **Bit 5** | **STO** | **Suspend time-out.** This bit will be set to a '1' when Suspend time-out occurs. An SMI will be generated if the associated SMI enable bit in the SMI Control register is set to a '1'. The software must write this bit to a '1' to deassert SMI#. |
| **Bit 4** | **HKT** | **House-keeping timeout detected.** This bit will be set to a '1' if the controller is in one of the house-keeping states and the house-keeping timer expires. An SMI will be generated if the associated SMI enable bit in the SMI Control register is set to a '1'. The software must write this bit to a '1' to deassert SMI#. If the SMI generation has been disabled, the hardware will automatically transition to doze or standby state. This bit then will be cleared on transition from Doze or Standby states to any other state. |
| **Bit 3** | **HKA** | **House-keeping activity detected.** This is a read-only bit and represents the OR of the System activity status registers masked (ANDed) with the corresponding bits in the House-keeping Activity enable registers. An SMI will be generated when this bit is a 1' and if the associated SMI enable bit in the SMI Control register is set to a '1'. The software can refer to Activity status register to determine the cause of the interrupt. The software must clear the corresponding bits of the Activity Status register to deassert SMI#. If SMI generation has been disabled and if the controller in Doze or Standby state, it will automatically transition to House-keeping state. |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 2** | **SAD** | **System Activity detected**. This is a read-only bit and represents the OR of the System activity Status registers masked (ANDed) with the corresponding bits of the System Activity enable registers. An SMI will be generated if this bit is a '1' and if the associated SMI enable bit in SMI Control register is set to a '1'. The software can refer to Activity status register to determine the cause of this interrupt. The software must clear the System Activity Status registers bits for the enabled system activities to deassert SMI#. If SMI generation has been disabled and if the controller is in Doze or Standby state, it will automatically transition to Power-on state. |
| **Bit 1** | **PID** | **Peripheral Inactivity detected.** This is a read-only bit and represents the OR of Peripheral Inactivity Status register bits masked (ANDed) with the associated Peripheral inactivity detection register bit. An SMI# will be generated when this bit is a '1'. The software can refer to Peripheral Inactivity status registers to determine which peripheral should be powered down. The software must clear the corresponding bits of the Peripheral Inactivity detection register to deassert SMI#. |
| **Bit 0** | **PAD** | **Peripheral Activity Detected.** This is a read-only bit and represents the OR of the System activity Status register masked (ANDed) with the corresponding bits of the Peripheral Activity detection registers. An SMI will be generated when this bit is a '1'. The software can refer to the System Activity status register to determine which peripheral caused the interrupt. The software must clear the corresponding bits of the System activity register to deassert SMI#. |

Programming notes:

The SMI# output is a logical OR of all the bits (ANDed with their respective SMI generation enable bits) in this register. SMI# output will be deasserted within 3 PCI clocks after the cause of the SMI# is cleared. This register defaults to 00h after reset deasserting SMI# output.

### *14.2.16. SMI STATUS REGISTER 1*

This register is similar to SMI Status register 0 in that it reports the cause of the SMI to the software.

*SMI_Stat1* Access = 0022h/0023h Regoffset = 074h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| S SMI | Rsv | | | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **S SMI** | **Software SMI.** This bit is set to a '1' by write writing a '1' in bit-1 of the SMI Control register. The software must clear this bit to deassert SMI#. |
| **Bits 6-0** | **Rsv** | **Reserved.** |

### *14.2.17. PERIPHERAL INACTIVITY STATUS REGISTER 0*

This register contains a '1' in a bit position if the associated peripheral was inactive for the entire duration of the last peripheral time-out period.

*Perif_Stat0* Access = 0022h/0023h Regoffset = 075h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PIO | SIO | KBD | FDC | HDC | ARO | Rsv | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **PIO** | **Parallel port (PIO)** activity. |
| **Bit 6** | **SIO** | **Serial port (SIO)** activity. |
| **Bit 5** | **KBD** | **Keyboard (KBD)** activity. |
| **Bit 4** | **FDC** | **Floppy Disk Controller (FDC)** activity. |
| **Bit 3** | **HDC** | **Hard Disk Controller (HDC)** activity. |
| **Bit 2** | **AR0** | **Address range 0** |
| **Bits 1-0** | **Rsv** | **Reserved.** |

It can also be cleared by software by writing a '1' in the bit which is set to '1'.

Programming notes:

A bit in this register is set to a '1' only at peripheral timer time-out. It is set to a '0' as soon as an activity from the associated peripheral is detected.

The status reflected in this register is not conditioned by whether or not the peripheral was enabled for in-activity detection through the Peripheral Inactivity Detection registers. The SMI however will be generated only if any of the enabled peripherals (via Peripheral Inactivity Enable register) were inactive for the entire duration of the peripheral time out.

*14.2.18. ACTIVITY STATUS REGISTER 0*

This register records presence of activity.

*Activ_Stat0* Access = 0022h/0023h Regoffset = 077h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DRQ | PCIM | PIO | SIO | KBD | FDC | HDC | Rsv |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **DRQ** | **DMA Request (DRQ)** activity. |
| **Bit 6** | **PCIM** | **PCI master device (PCIM)** activity. |
| **Bit 5** | **PIO** | **Parallel IO (PIO)** activity. |
| **Bit 4** | **SIO** | **Serial IO (SIO)** activity. |
| **Bit 3** | **KBD** | **Keyboard (KBD)** activity. |
| **Bit 2** | **FDC** | **Floppy Disk Controller (FDC)** activity. |
| **Bit 1** | **HDC** | **Hard Disk Controller (HDC)** activity. |
| **Bit 0** | **Rsv** | **Reserved.** |

Programming notes:

A ‘1’ in a bit position indicates that presence of the associated activity since the bit was last cleared. Once set, a bit of this register can only be cleared by software writing a ‘1’ to it or by reset or if auto power management is enabled then any transition to Doze or Standby state (including the ones from house-keeping states) will clear all enabled System and House-keeping activities.

The status reflected in this register is not conditioned by the settings of System Activity Enable, House-keeping Activity Enable, Peripheral Inactivity or Peripheral Activity Detection registers.

### *14.2.19. ACTIVITY STATUS REGISTER 1*

This register is similar to Activity Status register 0. It contains the status for the following bits.

*Activ_Stat1* Access = 0022h/0023h Regoffset = 078h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | AR0 | Rsv | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-6** | **Rsv** | **Reserved.** Must be programmed to ‘0’ |
| **Bit 5** | **AR0** | **Address range 0** |
| **Bits 4-0** | **Rsv** | **Reserved.** Must be programmed to ‘0’ |

### *14.2.20. ACTIVITY STATUS REGISTER 2*

This register is similar to Activity Status registers 0 and 1.

*Activ_Stat2* Access = 0022h/0023h Regoffset = 079h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IRQ15-1 | IRQ0 | NMI | Rsv | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **IRQ15-1** | **IRQ15-1** activity. |
| **Bit 6** | **IRQ0** | **IRQ0** activity. |
| **Bit 5** | **NMI** | **NMI** activity. |
| **Bits 4-0** | **Rsv** | **Reserved.** |

### *14.2.21. PMU STATUS REGISTER*

This register contains the state the power management controller currently is in.

*PMU* Access = 0022h/0023h Regoffset = 07Ah

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | PMU | PMU | PMU | PMU | PMU | PMU | PMU |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **Rsv** | **Reserved.** |
| **Bit 6** | **PMU** | **PMU microsecond clock test mode.** This bit is for factory use only and must be set to a '0' by the software. Setting this bit to a 1' causes the micro-second clock to tick at oscillator clock frequency instead of every microsec-ond. |
| **Bit 5** | **PMU** | **PMU millisecond clock test mode.** This bit is for factory use only and must be set to a '0' by the software. Setting this bit to a 1' causes the millisecond clock to tick at oscillator clock frequency instead of every millisecond. |
| **Bit 4** | **PMU** | **PMU second clock test mode.** This bit is for factory use only and must be set to a '0' by the software. Setting this bit to a 1' causes the second clock to tick at oscillator clock frequency instead of every second. |
| **Bit 3** | **PMU** | **PMU minute clock test mode.** This bit is for factory use only and must be set to a '0' by the software. Setting this bit to a 1' causes the minute clock to tick at oscillator clock frequency instead of every minute. |
| **Bit 2** | **PMU** | **PMU state** (see Table 14-7). |
| **Bit 1** | **PMU** | **PMU state** (see Table 14-7). |
| **Bit 0** | **PMU** | **PMU state** (see Table 14-7). |

**Table 14-7. PMU State**

| Bit 2 | Bit 1 | Bit 0 | PMU state |
|---|---|---|---|
| 0 | 0 | 0 | Power-on |
| 0 | 0 | 1 | Doze |
| 0 | 1 | 0 | Standby |
| 0 | 1 | 1 | Suspend |
| 1 | 0 | 1 | Doze_house_keeping |
| 1 | 1 | 0 | Standby_house_keeping |
| 1 | 1 | 1 | Reserved |

The architecture allows for: (1) the software to explicitly program the power-down state of the controller, or (2) the controller can change states automatically (auto-power down mode of operation), or (3) a mix of the two. Some power-down states are entered and exited automatically by the hardware, while others require software assist. This is based on the SMI Control register settings as follows:

Transition from Power-on to Doze state will take place automatically on Doze time-out, if bit-7 of the SMI control register is set to a ‘0’. Otherwise if bit-7 is programmed to be a ‘1’, an SMI will be generated and the software can change the state to Doze.

Transition from Doze to Power-on will take place automatically in the presence of an enabled system activity if bit-2 of the SMI control register is programmed to ’0’. Otherwise if bit-2 is programmed to a ‘1’, an SMI will be generated and software can change the state to Power-on.

Transition from Doze to Doze_house_keeping state will take place automatically if an enabled house_ keeping activity is detected and bit-4 of the SMI control register is set to a ‘0. Otherwise if bit-7 is programmed to be a ‘1’, an SMI will be generated.

Transition from Doze_house_keeping state to Doze will take place automatically on house-keeping time-out if bit-3 of the SMI control register is set to a ‘0’. Otherwise an SMI will be generated.

Transition from Doze_house_keeping state to Power-on will take place on detecting an enabled system activity automatically if bit-2 of the SMI control register is programmed to ‘0’. Otherwise an SMI will be generated.

Transitions from Doze or Power-on state to Standby will take place automatically on standby time-out if bit-6 of the SMI control register is set to a ‘0’. Otherwise an SMI will be generated.

Transition from Standby to Power-on will take place automatically in presence of an enabled system activity if bit-2 of the SMI control register is programmed to be a ‘0’. Otherwise if bit-2 is programmed to a ‘1’, an SMI will be generated and software can change the state to Power-on.

Transition from Standby to Standby_house_ keeping state will take place automatically if an enabled house_keeping activity is detected and bit-4 of the SMI control register is set to a ‘0’. Otherwise if bit-7 is programmed to be a ‘1’, an SMI will be generated.

Transition from Standby_house_keeping state to Standby will take place automatically on house-keeping time-out if bit-3 of the SMI control register is set to a ‘0’. Otherwise an SMI will be generated.

Transition from Standby_house_keeping state to Power-on will take place on detecting an enabled system activity automatically if bit-2 of the SMI control register is programmed to ‘0’. Otherwise SMI interrupt will be generated.

The hardware never transitions to Suspend state automatically.

The power saving features associated with each power-down state are independent of how the state was entered.

### *14.2.22. GENERAL PURPOSE REGISTER*

This is a read/write IO register that can be used by software.

*GP* Access = 0022h/0023h Regoffset = 07Bh

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| GP | GP | GP | GP | GP | GP | GP | GP |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **GP** | **General Purpose Register Bit 7.** |
| **Bit 6** | **GP** | **General Purpose Register Bit 6.** |
| **Bit 5** | **GP** | **General Purpose Register Bit 5.** |
| **Bit 4** | **GP** | **General Purpose Register Bit 4.** |
| **Bit 3** | **GP** | **General Purpose Register Bit 3.** |
| **Bit 2** | **GP** | **General Purpose Register Bit 2.** |
| **Bit 1** | **GP** | **General Purpose Register Bit 1.** |
| **Bit 0** | **GP** | **General Purpose Register Bit 0.** |

Programming notes:

Writing to this register also updates the external ‘373 latch that can be used to control external devices for power-down purposes. Reads of this register return the value of this internal register.

The GPIOCS# signal will be asserted when writing to this register to latch the data on the ISA data bus.

### *14.2.23. CLOCK CONTROL REGISTER 0*

This register allows control over power saving via stop clock modulation. The power-saving can be tuned to the power-management state the PMU is in.

*Clk_Cont0* Access = 0022h/0023h Regoffset = 07Ch

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| STPCLK | | | DSSS | | | STPCLK | Rsv |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-5** | **STPCLK** | **Power-on and housekeeping states** STPCLK# modulation control. These bits control the duty cycle of STPCLK# deassertion when the PMU is in Power-on or one of the house-keeping states given in Table 14-8.<br>The STPCLK# is deasserted and the duty-cycle control ignored if an SMI is pending. |
| **Bit 4-2** | **DSSS** | **Doze/Standby/Suspend states** STPCLK# modulation control. These bits control the duty cycle of the STPCLK# deassertion when PMU is in one of the power-down states as given in Table 14-9.<br>The STPCLK# is deasserted and the duty-cycle control ignored if an SMI is pending. |
| **Bit 1** | **STPCLK** | **STPCLK# modulation period.** If '1' then the period is 64ms else, if '0', then the period is 64µs. |
| **Bit 0** | **Rsv** | **Reserved.** |

**Table 14-8. Power-on and Housekeeping States**

| Bit 7 | Bit 6 | Bit 5 | Ratio | Power-on STPCLK# Modulation |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | STPCLK# is never asserted |
| 0 | 0 | 1 | 1/2 | 1 half period |
| 0 | 1 | 0 | 1/4 | 1 quarter period |
| 0 | 1 | 1 | 1/8 | one-eighth period |
| 1 | 0 | 0 | 1/16 | one-sixteenth period |
| 1 | 0 | 1 | 1/32 | 1/32 period |
| 1 | 1 | 0 | 1/64 | 1/64 period |
| 1 | 1 | 1 | | *Reserved.* |

**Table 14-9. Doze/Standby/Suspend States**

| Bit 4 | Bit 3 | Bit 2 | Ratio | Doze STPCLK# Modulation |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | STPCLK is never asserted |
| 0 | 0 | 1 | 1/2 | 1 half period |
| 0 | 1 | 0 | 1/4 | 1 quarter period |
| 0 | 1 | 1 | 1/8 | one-eighth period |
| 1 | 0 | 0 | 1/16 | one-sixteenth period |
| 1 | 0 | 1 | 1/32 | 1/32 period |
| 1 | 1 | 0 | 1/64 | 1/64 period |
| 1 | 1 | 1 | 0 | The entire period |

### *14.2.24. DOZE TIMER READ BACK REGISTER*

This read only register is provided for test purposes to read back the current value of the upper 8-bits of the 9-bit doze timer.

*Doze* Access = 0022h/0023h Regoffset = 088h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| Default value after reset = 00h | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-0** | | Bits 8-1 of the current value of the doze timer. |

Programming notes:

This register should not be used by the software.

Note that bit 0 of the current value of the doze timer is not readable.

### *14.2.25. STANDBY TIMER READ BACK REGISTER*

This read only register is provided for test purposes to read back the current value of 5-bit standby timer.

*Standby* Access = 0022h/0023h Regoffset = 089h

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | |
| Default value after reset = undefined | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-5** | **Rsv** | **Reserved**. |
| **Bits 4-0** | | Bits 4-0 of the current value of the standby timer. |

Programming notes:

This register should not be used by software.

### *14.2.26. SUSPEND TIMER READ BACK REGISTER*

This read only register is provided for test purposes to read back the current value of the 7-bit Suspend timer.

*Suspend* Access = 0022h/0023h Regoffset = 08Ah

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Rsv | | | | | | | |
| Default value after reset = undefined | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bit 7** | **Rsv** | **Reserved**. |
| **Bits 6-0** | | Bits 6-0 of the current value of the suspend timer. |

Programming notes:

This register should not be used by software.

### *14.2.27. HOUSE-KEEPING TIMER READ BACK REGISTER*

This read only register is provided for test purposes to read back the current value of the upper 8-bits of the 9-bit house-keeping timer.

*HK_Timer* Access = 0022h/0023h Regoffset = 08Bh

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| Default value after reset = undefined | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-0** | | Bits 8-1 of the house-keeping timer. |

Programming notes:

This register should not be used by software.

### *14.2.28. PERIPHERAL TIMER READ BACK REGISTER*

This read only register is provided for test purposes to read back the current value of the upper 8-bits of the 9-bit Peripheral timer.

*Perif_Timer* Access = 0022h/0023h Regoffset = 08Ch

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| Default value after reset = undefined | | | | | | | |

| Bit Number | Mnemonic | Description |
|---|---|---|
| **Bits 7-0** | | Bits 8-1 of the Peripheral timer. |

Programming notes:

This register should not be used by software.